

FOCUS for S/390

Developing Applications
Version 7.2

Cactus, EDA/SQL, FIDEL, FOCCALC, FOCUS, FOCUS Fusion, FOCUS Vision, Hospital-Trac, Information Builders, the Information Builders logo, Parlay, PC/FOCUS, SmartMart, SmartMode, SNAPpack, TableTalk, WALDO, Web390, WebFOCUS and WorldMART are registered trademarks and EDA, iWay, and iWay Software are trademarks of Information Builders, Inc.

Acrobat and Adobe are registered trademarks of Adobe Systems Incorporated.

Allaire and JRun are trademarks of Allaire Corporation.

NOMAD is a registered trademark of Aonix.

UniVerse is a registered trademark of Ardent Software, Inc.

IRMA is a trademark of Attachmate Corporation.

Baan is a registered trademark of Baan Company N.V.

SUPRA and TOTAL are registered trademarks of Cincom Systems, Inc.

Impromptu is a registered trademark of Cognos.

Alpha, DEC, DECnet, NonStop, and VAX are registered trademarks and Tru64, OpenVMS, and VMS are trademarks of Compaq Computer Corporation.

CA-ACF2, CA-Datcom, CA-IDMS, CA-Top Secret, and Ingres are registered trademarks of Computer Associates International, Inc.

MODEL 204 and M204 are registered trademarks of Computer Corporation of America.

Paradox is a registered trademark of Corel Corporation.

StorHouse is a registered trademark of FileTek, Inc.

HP MPE/iX is a registered trademark of Hewlett Packard Corporation.

Informix is a registered trademark of Informix Software, Inc.

ACF/VTAM, AIX, AS/400, CICS, DB2, DRDA, Distributed Relational Database Architecture, IBM, MQSeries, MVS/ESA, OS/2, OS/390, OS/400, RACF, RS/6000, S/390, VM/ESA, VSE/ESA and VTAM are registered trademarks and DB2/2, Hyperspace, IMS, MVS, QMF, SQL/DS, WebSphere, z/OS and z/VM are trademarks of International Business Machines Corporation.

INTERSOLVE and Q+E are registered trademarks of INTERSOLVE.

Orbix is a registered trademark of Iona Technologies Inc.

Approach and DataLens are registered trademarks of Lotus Development Corporation.

ObjectView is a trademark of Matesys Corporation.

ActiveX, FrontPage, Microsoft, MS-DOS, PowerPoint, Visual Basic, Visual C++, Visual FoxPro, Windows, and Windows NT are registered trademarks of Microsoft Corporation.

Teradata is a registered trademark of NCR International, Inc.

Netscape, Netscape FastTrack Server, and Netscape Navigator are registered trademarks of Netscape Communications Corporation.

CORBA is a trademark of Object Management Group, Inc.

Oracle is a registered trademark and Rdb is a trademark of Oracle Corporation.

PeopleSoft is a registered trademark of PeopleSoft, Inc.

INFOAccess is a trademark of Pioneer Systems, Inc.

Progress is a registered trademark of Progress Software Corporation.

Red Brick Warehouse is a trademark of Red Brick Systems.

SAP and SAP R/3 are registered trademarks and SAP Business Information Warehouse and SAP BW are trademarks of SAP AG.

Silverstream is a trademark of Silverstream Software.

ADABAS is a registered trademark of Software A.G.

CONNECT:Direct is a trademark of Sterling Commerce.

Java and all Java-based marks, NetDynamics, Solaris, SunOS, and iPlanet are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

PowerBuilder and Sybase are registered trademarks and SQL Server is a trademark of Sybase, Inc.

Unicode is a trademark of Unicode, Inc.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Due to the nature of this material, this document refers to numerous hardware and software products by their trade names. In most, if not all cases, these designations are claimed as trademarks or registered trademarks by their respective companies. It is not this publisher's intent to use any of these names generically. The reader is therefore cautioned to investigate all claimed trademark rights before using any of these names other than to refer to the product described.

Copyright © 2001 by Information Builders, Inc. All rights reserved. This manual, or parts thereof, may not be reproduced in any form without the written permission of Information Builders, Inc.

Printed in the U.S.A.

Preface

This documentation describes how to create the metadata for the data sources that your FOCUS procedures will access in FOCUS® Version 7.2. It is intended for database administrators, application developers, or other information technology professionals who will create the metadata used by FOCUS to access corporate data. This manual is part of the FOCUS for S/390 documentation set.

References to MVS apply to all supported versions of the OS/390, z/OS™, and MVS operating environments. References to VM apply to all supported versions of the VM/ESA and z/VM™ operating environments.

The documentation set consists of the following components:

- The *Creating Reports* manual describes FOCUS Reporting environments and features.
- The *Describing Data* manual explains how to create the metadata for the data sources that your FOCUS procedures will access.
- The *Developing Applications* manual describes FOCUS Application Development tools and environments.
- The *Maintaining Databases* manual describes FOCUS data management facilities and environments.
- The *Using Functions* manual describes internal functions and user-written subroutines.
- The *Overview and Operating Environments* manual contains an introduction to FOCUS and FOCUS tools and describes how to use FOCUS in the VM/CMS and MVS (OS/390) environments.

The users' documentation for FOCUS Version 7.2 is organized to provide you with a useful, comprehensive guide to FOCUS.

Chapters need not be read in the order in which they appear. Though FOCUS facilities and concepts are related, each chapter fully covers its respective topic. To enhance your understanding of a given topic, references to related topics throughout the documentation set are provided. The following pages detail documentation organization and conventions.

How This Manual Is Organized

This manual is organized as follows:

Chapter/Appendix		Contents
1	<i>Customizing Your Environment</i>	Describes how to control your FOCUS environment with the SET command.
2	<i>Querying Your Environment</i>	Describes how to use query commands to retrieve information about the FOCUS environment.
3	<i>Managing an Application With Dialogue Manager</i>	Describes how to make a report procedure more dynamic using Dialogue Manager commands.
4	<i>Defining a Word Substitution</i>	Describes how to define a string substitution that can be used in a report request.
5	<i>Enhancing Application Performance</i>	Describes FOCUS facilities for increasing the speed of your application.
6	<i>Working With Cross-Century Dates</i>	Describes techniques for assigning a century date to dates with two-digit years.
7	<i>Euro Currency Support</i>	Describes how to perform currency conversions according to the rules established by the European Union.
8	<i>Designing Windows With Window Painter</i>	Describes how to create FOCUS windows and menus that work in conjunction with a procedure.
A	<i>Master Files and Diagrams</i>	Contains Master Files and diagrams of same data sources used in documentation examples.
B	<i>Error Messages</i>	Describes how to obtain additional information about error messages in FOCUS.

Summary of New Features

The new FOCUS features and enhancements described in this documentation set are listed in the following table.

New Feature	Manual	Chapter
Field-based Reformatting	<i>Creating Reports</i>	Chapter 1, <i>Creating Tabular Reports</i>
Increased Report Width	<i>Creating Reports</i>	Chapter 1, <i>Creating Tabular Reports</i>
ACROSS-TOTAL	<i>Creating Reports</i>	Chapter 4, <i>Sorting Tabular Reports</i>
Tiles	<i>Creating Reports</i>	Chapter 4, <i>Sorting Tabular Reports</i>
DEFINE FILE SAVE and DEFINE FILE RETURN	<i>Creating Reports</i>	Chapter 6, <i>Creating Temporary Fields</i>
Forecast	<i>Creating Reports</i>	Chapter 6, <i>Creating Temporary Fields</i>
Creating Comma-Delimited Files	<i>Creating Reports</i>	Chapter 11, <i>Saving and Reusing Report Output</i>
Creating Tab-Delimited Files	<i>Creating Reports</i>	Chapter 11, <i>Saving and Reusing Report Output</i>
Long Master File Names	<i>Creating Reports</i>	Chapter 11, <i>Saving and Reusing Report Output</i>
JOIN WHERE	<i>Creating Reports</i>	Chapter 13, <i>Joining Data Sources</i>
KEEPDEFINES	<i>Creating Reports</i>	Chapter 13, <i>Joining Data Sources</i>
Long Master File Names	<i>Describing Data</i>	Chapter 1, <i>Understanding a Data Source Description</i>
4K Alpha Fields	<i>Describing Data</i>	Chapter 4, <i>Describing an Individual Field</i>
Extended Currency Symbol Support	<i>Describing Data</i>	Chapter 4, <i>Describing an Individual Field</i>
SUFFIX = COMT/COMMA/TABT	<i>Describing Data</i>	Chapter 5, <i>Describing a Sequential, VSAM, or ISAM Data Source</i>

New Feature	Manual	Chapter
AUTODATE	<i>Describing Data</i>	Chapter 6, <i>Describing a FOCUS Data Source</i>
CDN parameter	<i>Developing Applications</i>	Chapter 1, <i>Customizing Your Environment</i>
CENT-ZERO parameter	<i>Developing Applications</i>	Chapter 1, <i>Customizing Your Environment</i>
ERROROUT parameter	<i>Developing Applications</i>	Chapter 1, <i>Customizing Your Environment</i>
KEEPDEFINES parameter	<i>Developing Applications</i>	Chapter 1, <i>Customizing Your Environment</i>
PCOMMA parameter	<i>Developing Applications</i>	Chapter 1, <i>Customizing Your Environment</i>
Unlimited nested -INCLUDE commands	<i>Developing Applications</i>	Chapter 3, <i>Managing an Application With Dialogue Manager</i>
SQUEEZ function	<i>Using Functions</i>	Chapter 3, <i>Character Functions</i>
STRIP function	<i>Using Functions</i>	Chapter 3, <i>Character Functions</i>
TRIM function	<i>Using Functions</i>	Chapter 3, <i>Character Functions</i>
DYNAM ALLOC LONGNAME	<i>Overview and Operating Environments</i>	Chapter 5, <i>OS/390 and MVS Guide to Operations</i>

Documentation Conventions

The following conventions apply throughout this manual:

Convention	Description
<code>THIS TYPEFACE</code>	Denotes a command that you must enter in uppercase, exactly as shown.
<i>this typeface</i>	Denotes a value that you must supply.
{ }	Indicates two choices from which you must choose one. You type one of these choices, not the braces.
	Separates two mutually exclusive choices in a syntax line. Type one of these choices, not the symbol.
[]	Indicates optional parameters. None of them is required, but you may select one of them. Type only the information within the brackets, not the brackets.
<u>underscore</u>	Indicates the default value.
...	Indicates that you can enter a parameter multiple times. Type only the information, not the ellipsis points.
. . . .	Indicates that there are (or could be) intervening or additional commands.

Related Publications

See the Information Builders Publications Catalog for the most up-to-date listing and prices of technical publications, plus ordering information. To obtain a catalog, contact the Publications Order Department at (800) 969-4636.

You can also visit our World Wide Web site, <http://www.informationbuilders.com>, to view a current listing of our publications and to place an order.

Customer Support

Do you have questions about FOCUS?

Call Information Builders Customer Support Services (CSS) at (800) 736-6130 or (212) 736-6130. Customer Support Consultants are available Monday through Friday between 8:00 a.m. and 8:00 p.m. EST to address all your FOCUS questions. Information Builders consultants can also give you general guidance regarding product capabilities and documentation. Please be ready to provide your six-digit site code number (xxxx.xx) when you call.

You can also access support services electronically, 24 hours a day, with InfoResponse Online. InfoResponse Online is accessible through our World Wide Web site, <http://www.informationbuilders.com>. It connects you to the tracking system and known-problem database at the Information Builders support center. Registered users can open, update, and view the status of cases in the tracking system, and read descriptions of reported software issues. New users can register immediately for this service. The technical support section of www.informationbuilders.com also provides usage techniques, diagnostic tips, and answers to frequently asked questions.

To learn about the full range of available support services, ask your Information Builders representative about InfoResponse Online, or call (800) 969-INFO.

Information You Should Have

To help our consultants answer your questions most effectively, be ready to provide the following information when you call:

- Your six-digit site code number (xxxx.xx).
- The FOCEXEC procedure (preferably with line numbers).
- Master File with picture (provided by CHECK FILE).
- Run sheet (beginning at login, including call to FOCUS), containing the following information:
 - ? RELEASE
 - ? FDT
 - ? LET
 - ? LOAD
 - ? COMBINE
 - ? JOIN
 - ? DEFINE
 - ? STAT
 - ? SET
 - ? SET GRAPH
 - ? USE
 - For MVS, ? TSO DDNAME
 - For VM, CMS QFI

- The exact nature of the problem:
 - Are the results or the format incorrect; are the text or calculations missing or misplaced?
 - The error message and code, if applicable.
 - Is this related to any other problem?
- Has the procedure or query ever worked in its present form? Has it been changed recently? How often does the problem occur?
- What release of the operating system are you using? Has it, FOCUS, your security system, or an interface system changed?
- Is this problem reproducible? If so, how?
- Have you tried to reproduce your problem in the simplest form possible? For example, if you are having problems joining two databases, have you tried executing a query containing just the code to access the database?
- Do you have a trace file?
- How is the problem affecting your business? Is it halting development or production? Do you just have questions about functionality or documentation?

User Feedback

In an effort to produce effective documentation, the Documentation Services staff at Information Builders welcomes any opinion you can offer regarding this manual. Please use the Reader Comments form at the end of this manual to relay suggestions for improving the publication or to alert us to corrections. You can also use the Document Enhancement Request Form on our Web site, <http://www.informationbuilders.com>.

Thank you, in advance, for your comments.

Information Builders Consulting and Training

Interested in training? Information Builders Education Department offers a wide variety of training courses for this and other Information Builders products.

For information on course descriptions, locations, and dates, or to register for classes, visit our World Wide Web site (<http://www.informationbuilders.com>) or call (800) 969-INFO to speak to an Education Representative.

Contents

1	Customizing Your Environment	1-1
	The SET Command	1-2
	SET Parameter Syntax.....	1-3
2	Querying Your Environment	2-1
	Using Query Commands	2-2
	Displaying Combined Structures.....	2-3
	Displaying Virtual Fields	2-4
	Displaying the Currency Data Source in Effect.....	2-5
	Displaying Available Fields	2-5
	Displaying the File Directory Table	2-6
	Displaying Field Information for a Master File.....	2-8
	Displaying Data Source Statistics.....	2-9
	Displaying Defined Functions	2-11
	Displaying HiperBudget Limits and Usage.....	2-11
	Displaying HOLD Fields.....	2-12
	Displaying JOIN Structures.....	2-13
	Displaying National Language Support.....	2-14
	Displaying LET Substitutions	2-14
	Displaying Information About Loaded Files.....	2-15
	Displaying Explanations of Error Messages.....	2-15
	Displaying PF Key Assignments	2-16
	Querying Which PTFs Have Been Applied for a Specific Release	2-16
	Displaying the Release Number	2-17
	Displaying Parameter Settings.....	2-18
	Displaying Parameters That Cannot Be Set in an Area.....	2-20
	Displaying Graph Parameters.....	2-21
	Displaying Command Statistics.....	2-22
	Displaying StyleSheet Parameter Settings.....	2-24
	Displaying Information About the SU Machine.....	2-26
	Displaying Data Sources Specified With USE.....	2-26
	Displaying Global Variable Values.....	2-27

3	Managing an Application With Dialogue Manager.....	3-1
	Overview of Dialogue Manager Capabilities	3-2
	Overview of Dialogue Manager Variables	3-5
	Creating and Storing Procedures	3-6
	Executing Procedures	3-7
	Controlling Access to Data.....	3-7
	Including Comments in a Procedure	3-8
	Overview of Dialogue Manager Commands	3-9
	Sending a Message to the User: -TYPE	3-11
	Controlling Execution: -RUN, -EXIT, and -QUIT.....	3-12
	Executing Stacked Commands and Continuing the Procedure: -RUN.....	3-12
	Executing Stacked Commands and Exiting the Procedure: -EXIT	3-13
	Canceling Execution of the Procedure: -QUIT	3-14
	Branching	3-15
	-GOTO Processing	3-15
	Compound -IF Tests.....	3-18
	Using Operators and Functions in -IF Tests.....	3-19
	Screening Values With -IF Tests.....	3-19
	Testing the Status of a Query	3-23
	Looping	3-24
	Ending a Loop	3-26
	Using Expressions: -SET.....	3-26
	Computing a New Variable.....	3-27
	Using the DECODE Function	3-28
	Using the EDIT Function	3-29
	Using the TRUNCATE Function	3-30
	Controlling a Loop With -SET.....	3-32
	Setting a Date	3-33
	Calling a Function	3-33
	Additional Facilities	3-36
	Establishing Startup Conditions	3-36
	Incorporating Multiple Procedures.....	3-37
	Nesting Procedures With -INCLUDE	3-40
	Using EXEC.....	3-41
	Developing an Open-Ended Procedure	3-41
	Debugging With &ECHO	3-42
	Testing Dialogue Manager Command Logic With &STACK	3-43
	Locking Procedure Users Out of FOCUS	3-44
	Writing to Files: -WRITE.....	3-45

Using Variables in Procedures	3-49
Querying the Values of Variables	3-51
Local Variables	3-52
Global Variables.....	3-53
System Variables.....	3-54
Statistical Variables.....	3-59
Special Variables.....	3-61
Using Variables to Alter Commands.....	3-62
Evaluating a Variable Immediately	3-62
Concatenating Variables.....	3-64
Creating an Indexed Variable.....	3-64
Supplying Values for Variables at Run Time.....	3-66
Supplying Values Without Prompting.....	3-68
Supplying Values With -DEFAULTS.....	3-70
Supplying Values With -SET	3-71
Supplying Values With -READ	3-72
Direct Prompting With -PROMPT	3-73
Full-Screen Data Entry With -CRTFORM.....	3-75
Selecting Data From Menus and Windows With -WINDOW	3-75
Implied Prompting.....	3-75
Verifying Input Values.....	3-76
Dialogue Manager Quick Reference	3-80
System Defaults and Limits	3-97
4 Defining a Word Substitution.....	4-1
The LET Command.....	4-2
Variable Substitution	4-5
Null Substitution.....	4-7
Multiple-line Substitution.....	4-8
Recursive Substitution.....	4-8
Using a LET Substitution in a COMPUTE or DEFINE Command	4-9
Checking Current LET Substitutions	4-10
Interactive LET Query: LET ECHO	4-10
Clearing LET Substitutions	4-11
Saving LET Substitutions in a File.....	4-12
Assigning Phrases to Function Keys	4-12

5	Enhancing Application Performance.....	5-1
	FOCUS Facilities	5-2
	Loading a File.....	5-2
	Loading Master Files, FOCUS Procedures, and Access Files.....	5-3
	Loading a Compiled MODIFY Request.....	5-5
	Loading a MODIFY Request	5-5
	Displaying Information About Loaded Files.....	5-6
	Compiling a MODIFY Request.....	5-7
	Accessing a FOCUS Data Source (MVS Only)	5-8
	Using MINIO	5-9
	Determining if a Previous Command Used MINIO	5-10
	Enhancing File Management With HiperFOCUS	5-12
	Activating HiperFOCUS	5-13
	Installing and Configuring HiperFOCUS.....	5-13
	Installing HiperBudget on OS/390	5-14
	Creating Temporary Files in Hiperspaces With HiperFile on OS/390.....	5-16
	Creating a Temporary Sort File in Hiperspace on CMS.....	5-19
	Creating Cache in Hiperspaces on OS/390.....	5-19
	Controlling HiperFOCUS Use With the HiperRule Facility	5-21
6	Working With Cross-Century Dates	6-1
	When Do You Use the Sliding Window Technique?.....	6-2
	The Sliding Window Technique.....	6-2
	Defining a Sliding Window.....	6-3
	Creating a Dynamic Window Based on the Current Year.....	6-4
	Applying the Sliding Window Technique	6-5
	When to Supply Settings for DEFCENT and YRTHRESH	6-5
	Date Validation	6-6
	Defining a Global Window With SET.....	6-7
	Defining a Dynamic Global Window With SET	6-10
	Querying the Current Global Value of DEFCENT and YRTHRESH.....	6-12
	Defining a File-Level or Field-Level Window in a Master File.....	6-13
	Defining a Window for a Virtual Field.....	6-19
	Defining a Window for a Calculated Value	6-25
	Additional Support for Cross-Century Dates	6-30
7	Euro Currency Conversion.....	7-1
	Integrating the Euro Currency	7-2
	Converting Currencies.....	7-2
	Creating the Currency Data Source	7-4

Identifying Fields That Contain Currency Data	7-6
Activating the Currency Data Source	7-8
Processing Currency Data	7-9
Querying the Currency Data Source in Effect	7-14
8 Designing Windows With Window Painter.....	8-1
Introduction	8-2
How Do Window Applications Work?	8-3
Window Files and Windows.....	8-4
Types of Windows You Can Create	8-4
Creating Windows.....	8-14
Integrating Windows and the FOCEXEC.....	8-21
Transferring Control in Window Applications.....	8-22
Return Values.....	8-24
Goto Values.....	8-25
Window System Variables	8-26
Testing Function Key Values	8-26
Executing a Window From the FOCUS Prompt	8-28
Tutorial: A Menu-Driven Application.....	8-29
Creating the Application FOCEXEC	8-31
Creating the Window File	8-33
Executing the Application.....	8-51
Window Painter Screens.....	8-51
Invoking Window Painter	8-52
Entry Menu.....	8-53
Main Menu	8-54
Window Creation Menu	8-57
Window Design Screen	8-59
Window Options Menu	8-61
Utilities Menu.....	8-72
Transferring Window Files.....	8-75
Creating a Transfer File.....	8-76
Transferring the File to the New Environment.....	8-77
Editing the Transfer File.....	8-77
Compiling the Transfer File	8-83
A Master Files and Diagrams.....	A-1
Creating Sample Data Sources	A-2
The EMPLOYEE Data Source	A-3
The EMPLOYEE Master File	A-4
The EMPLOYEE Structure Diagram	A-5

The JOBFILE Data Source.....	A-6
The JOBFILE Master File.....	A-6
The JOBFILE Structure Diagram.....	A-6
The EDUCFILE Data Source.....	A-7
The EDUCFILE Master File	A-7
The EDUCFILE Structure Diagram.....	A-7
The SALES Data Source	A-8
The SALES Master File	A-8
The SALES Structure Diagram.....	A-9
The PROD Data Source.....	A-10
The PROD Master File.....	A-10
The PROD Structure Diagram.....	A-10
The CAR Data Source	A-11
The CAR Master File	A-11
The CAR Structure Diagram.....	A-12
The LEDGER Data Source	A-13
The LEDGER Master File.....	A-13
The LEDGER Structure Diagram	A-13
The FINANCE Data Source.....	A-14
The FINANCE Master File	A-14
The FINANCE Structure Diagram.....	A-14
The REGION Data Source	A-15
The REGION Master File	A-15
The REGION Structure Diagram	A-15
The COURSES Data Source	A-16
The COURSES Master File	A-16
The COURSES Structure Diagram	A-16
The EMPDATA Data Source	A-17
The EMPDATA Master File	A-17
The EMPDATA Structure Diagram.....	A-17
The EXPERSON Data Source.....	A-18
The EXPERSON Master File.....	A-18
The EXPERSON Structure Diagram.....	A-18
The TRAINING Data Source	A-19
The TRAINING Master File	A-19
The TRAINING Structure Diagram.....	A-19
The PAYHIST File.....	A-20
The PAYHIST Master File.....	A-20
The PAYHIST Structure Diagram	A-20

The COMASTER File	A-21
The COMASTER Master File	A-22
The COMASTER Structure Diagram	A-23
The VideoTrk and MOVIES Data Sources	A-24
VideoTrk Master File	A-24
MOVIES Master File	A-24
VideoTrk Structure Diagram	A-25
MOVIES Structure Diagram	A-26
The VIDEOTR2 Data Source	A-26
The VIDEOTR2 Master File	A-26
The VIDEOTR2 Access File	A-27
The VIDEOTR2 Structure Diagram	A-28
The Gotham Grinds Data Sources	A-29
The GGDEMOG Data Source	A-29
The GGORDER Data Source	A-30
The GGPRODS Data Source	A-31
The GGSALES Data Source	A-32
The GGSTORES Data Source	A-33
B Error Messages	B-1
Accessing Error Files	B-2
Displaying Messages Online	B-3
Index	I-1

CHAPTER 1

Customizing Your Environment

Topics:

- The SET Command
- SET Parameter Syntax

The SET command enables you to change parameters that govern your FOCUS environment. These parameters control output, work areas, the Hot Screen facility and other FOCUS features.

The SET Command

The SET command enables you to customize both the application development and runtime environment. It controls the way that reports and graphs display on the screen or printer; the content of reports and graphs; data retrieval characteristics that affect performance; and system responses to end user requests.

Syntax

How to Set Parameters

```
SET parameter = option[, parameter = option,...]
```

where:

parameter

Is the FOCUS setting you wish to change.

option

Is one of a number of options available for each parameter.

You can set several parameters in one command by separating each with a comma.

You may include as many parameters as you can fit on one line. Repeat the SET keyword for each new line.

Syntax

How to Set Parameters in a Request

Many SET parameters that change system defaults can be issued from TABLE and GRAPH requests. SET used in this manner is temporary, affecting only the current request. The syntax is

```
ON {TABLE|GRAPH} SET parameter value [AND parameter value ...]
```

where:

parameter

Is the system default you wish to change.

value

Is an acceptable value that will replace the default value.

Example**Setting Parameters in a Request**

For example,

```
TABLE FILE EMPLOYEE
PRINT CURR_SAL BY EMP_ID
ACROSS DEPARTMENT
ON TABLE SET NODATA NONE
END
```

changes the default NODATA character for missing data from a period to the word NONE.

SET parameters that cannot be issued from within TABLE include ASNAMES, BINS, and HOLDATTR. You can generate a list of such parameters by issuing the following command:

```
? SET NOT ONTABLE
```

SET Parameter Syntax

This topic alphabetically lists the SET parameters that control the environment with a description and their syntax.

Parameter: ACCBLN

Description: Accepts blank or zero values for fields with ACCEPT commands in the Master File (see the *Describing Data* manual).

Syntax: SET ACCBLN = {ON|OFF}

where:

ON

Accepts blank and zero values for fields with ACCEPT commands unless blank or zero values are explicitly coded in the list of acceptable values. This value is the default.

OFF

Does not accept blank and zero values for fields with ACCEPT commands unless blank or zero values are explicitly coded in the list of acceptable values.

Parameter:	<code>AGGR[RATIO]</code>
Description:	Determines the ratio of aggregation based on retrieved records and the final size of the answer set.
Syntax:	<code>SET AGGR[RATIO] = {<i>n</i> <u>9</u>}</code> where: <code><i>n</i></code> Is the ratio of aggregation. The default value is 9.
Parameter:	<code>ALL</code>
Description:	Handles missing segment instances in a report.
Syntax:	<code>SET ALL = {ON <u>OFF</u> PASS}</code> where: <code>ON</code> Includes missing segment instances in a report when fields in the segment are not screened by WHERE or IF criteria in the request. The missing field values are denoted by the NODATA character, set with the NODATA parameter (see NODATA). <code><u>OFF</u></code> Omits missing segment instances from a report. This value is the default. <code>PASS</code> Includes missing segment instances in a report regardless of WHERE or IF criteria in the request. This option is not supported when MULTIPATH = COMPOUND (see MULTIPATH).

Parameter:	ALLOWCVTERR
Description:	<p>This parameter applies to non-FOCUS data sources when converting from the way the date is stored (ACTUAL attribute) to the way it is formatted (FORMAT or USAGE attribute).</p> <p>Controls the display of a row of data that contains an invalid date format (formerly called a smart date). When it is set to ON, the invalid date format is returned as the base date or a blank, depending on the settings for the MISSING and DATEDISPLAY parameters.</p>
Syntax:	<p>SET ALLOWCVTERR = {ON <u>OFF</u>}</p> <p>where:</p> <p>ON</p> <p>Displays a row of data that contains an invalid date format. When ALLOWCVTERR is set to ON, the display of invalid dates is determined by the settings of the MISSING attribute and DATEDISPLAY command.</p> <ul style="list-style-type: none">• If DATEDISPLAY and MISSING are set to OFF, a blank is returned.• If DATEDISPLAY is set to OFF, and MISSING is set to ON, the value of the NODATA character (a period, by default) is returned (see NODATA).• If DATEDISPLAY and MISSING are set to ON, the value of the NODATA character (a period, by default) is returned.• If DATEDISPLAY is set to ON, and MISSING is set to OFF, the base date is returned (either December 31, 1900, for dates with YMD or YYMD format; or January 1901, for dates with YM, YYM, YQ, or YYQ format). <p>OFF</p> <p>Does not display a row of data that contains an invalid date format and generates an error message. This value is the default.</p>

Parameter:	ASNAMES
Description:	Controls the FIELDNAME attribute in a HOLD Master File. When an AS phrase is used in a TABLE request, the specified literal is used as a field name in a HOLD file. Also controls how field names are specified for the values of an ACROSS field when a HOLD file is created.
Syntax:	<pre>SET ASNAMES = {ON OFF FOCUS}</pre> <p>where:</p> <p>ON</p> <p>Uses the AS phrase for the field name, and controls the way ACROSS fields are named in HOLD files in any format.</p> <p>OFF</p> <p>Does not use the AS phrase for the field name, or affect the way ACROSS fields are named.</p> <p>FOCUS</p> <p>Uses the AS phrase for the field name, and controls the way ACROSS fields are named in HOLD files only in FOCUS format. This value is the default.</p>
Parameter:	AUTOINDEX
Description:	Retrieves data faster by automatically taking advantage of indexed fields in most cases where TABLE requests contain equality or range tests on those fields. Applies only to FOCUS data sources.
	AUTOINDEX is never performed when the TABLE request contains an alternate file view, for example, TABLE FILE <i>filename.filename</i> . Indexed retrieval is not performed when the TABLE request contains BY HIGHEST or BY LOWEST phrases and AUTOINDEX is ON.
Syntax:	<pre>SET AUTOINDEX = {ON OFF}</pre> <p>where:</p> <p>ON</p> <p>Uses indexed retrieval when possible.</p> <p>OFF</p> <p>Uses indexed retrieval only when explicitly specified via an indexed view, for example, TABLE FILE <i>filename.fieldname</i>. This value is the default.</p>

Parameter:	AUTOPATH
Description:	Dynamically selects an optimal retrieval path for accessing a FOCUS data source by analyzing the data source structure and the fields referenced, and choosing the lowest possible segment as the entry point. Use AUTOPATH only if your field is not indexed.
Syntax:	<code>SET AUTOPATH = {ON OFF}</code> where: ON Dynamically selects an optimal retrieval path. This value is the default. OFF Uses sequential data retrieval. The end user controls the retrieval path through <i>filename.segname</i> .
Parameter:	AUTOSTRATEGY
Description:	Determines when FOCUS stops the search for a key field specified in a WHERE or IF test. When set to ON, the search ends when the key field is found, optimizing retrieval speed. When set to OFF, the search continues to the end of the data source.
Syntax:	<code>SET AUTOSTRATEGY = {ON OFF}</code> where: ON Stops the search when a match is found. This value is the default. OFF Searches the entire data source.
Parameter:	AUTOTABLEF
Description:	Avoids creating the internal matrix based on the features used in the query. Avoiding internal matrix creation reduces internal overhead costs and yields better performance.
Syntax:	<code>SET AUTOTABLEF = {ON OFF}</code> where: ON Does not create an internal matrix. This value is the default. OFF Creates an internal matrix.

Parameter:	BINS
Description:	Specifies the number of pages of memory (blocks of 4,096 bytes) used for data source buffers. You can vary BINS from 13 to 64 pages. The default is 64, and this is the recommended value.
Syntax:	<code>SET BINS = <i>n</i></code> where: <i>n</i> Is the number of pages used for data source buffers. Valid values are 13 to 64. The default value is 64.
Parameter:	BLKCALC
Description:	This parameter applies only to MVS. Enables system-determined blocking for HOLD files written to DASD; files written to tape have BLKSIZE 32760, the operating-system maximum. The SET BLKCALC command must be issued before the TABLE request and cannot be set within a request.
Syntax:	<code>SET BLKCALC = {<u>NEW</u> OLD}</code> where: <u>NEW</u> Calculates optimal blocking factors for both 3380 and 3390 device types. This value is the default. <u>OLD</u> Uses the method of calculating BLKSIZE that was used prior to FOCUS Release 6.8.
Parameter:	BOTTOMMARGIN
Description:	This command applies to StyleSheets. Sets the bottom boundary, in inches, of report contents on a page.
Syntax:	<code>SET BOTTOMMARGIN = {<i>n</i>/<u>.25</u>}</code> where: <i>n</i> Is the bottom margin, in inches, for report contents on a page. The default is .25 inches.

Parameter:	BUSDAYS
Description:	Specifies which days are considered business days and which days are not if your business does not follow the traditional Monday through Friday week.
Syntax:	<code>SET BUSDAYS = week</code> where: <code>week</code> Is SMTWTFS, representing the days of the week. Any day that you do not wish to designate as a business day must be replaced with an underscore in that day's designated place.
Parameter:	BYPANEL
Description:	This parameter applies only to HOTSCREEN. Controls the repetition of BY fields on panels. When BYPANEL is specified, the maximum number of panels is 99. When BYPANEL is OFF, the maximum number of panels is four.
Syntax:	<code>SET BYPANEL = option</code> where: <code>option</code> Is one of the following: <code>ON</code> repeats BY field values on panels. <code>OFF</code> does not repeat field values on panels. <code>0</code> does not divide columns between panels. <code>n</code> repeats <i>n</i> columns on each panel.
Parameter:	BYSCROLL
Description:	This parameter applies only to HOTSCREEN. Scrolls report headings and footers scroll along with the report contents.
Syntax:	<code>SET {BYSCROLL BYPANELSCROL} = {ON OFF}</code> where: <code>ON</code> Scrolls report headings and footings along with report contents. <code>OFF</code> Does not scroll report headings and footings along with report contents.

Parameter: `CACHE`

Description:

Stores 4K FOCUS data source pages in memory and buffers them between the data source and BINS.

When a procedure calls for a read of a data source page, FOCUS first searches BINS, then cache memory, and then the data source on disk. If the page is found in cache, FOCUS does not have to perform an I/O to disk.

When a procedure calls for a write of a data source page, the page is written from BINS to disk. The updated page is also copied into cache memory so that the cache and disk versions remain the same. Unlike reads, cache memory does not save disk I/Os for write procedures.

FOCSORT pages are also written to cache; when the cache becomes full, they are written to disk. For optimal results, set cache to hold the entire data source plus the size of FOCSORT for the request. To estimate the size of FOCSORT for a given request, issue the ? STAT command (discussed in Chapter 2, *Querying Your Environment*, then add the number of SORTPAGES listed to the number of data source pages in memory. Issue a SET CACHE command for that amount. If cache is set to 50, 50 4K pages of contiguous storage are allocated to cache. The maximum number of cache pages can be set at installation.

To clear the CACHE setting, issue a SET CACHE = *n* command. This command flushes the buffer; that is, everything in cache memory is lost.

Syntax:

`SET CACHE = {0|n}`

where:

0

Allocates no space to cache; cache is inactive. This value is the default.

n

Is the number of 4K pages of contiguous storage allocated to cache memory. The minimum is two pages; the maximum is determined by the amount of memory available. If HiperFOCUS is activated, the default cache size is 256 pages (1MB) and the cache is placed in a hyperspace.

Parameter:	<code>CARTESIAN</code>
Description:	<p>Applies to requests containing PRINT or LIST.</p> <p>Generates a report containing all combinations of non-related data instances in the case of a multi-path request. ACROSS cancels this parameter.</p>
Syntax:	<p><code>SET CARTESIAN = {ON OFF}</code></p> <p>where:</p> <p><code>ON</code></p> <p>Generates a report with non-related records.</p> <p><code>OFF</code></p> <p>Disables the Cartesian product. This value is the default.</p>
Parameter:	<code>CDN</code>
Description:	<p>Specifies punctuation used in numerical notation.</p> <p>Continental Decimal Notation (CDN) is supported for output in TABLE requests. It is not supported in DEFINE or COMPUTE commands.</p>
Syntax:	<p><code>SET CDN = option</code></p> <p>where:</p> <p><code>option</code></p> <p>Is one of the following:</p> <p><code>ON</code> sets the decimal separator as a comma and the thousands separator as a period. For example, the number 3,045,000.76 is represented as 3.045.000,76. ON should be used for Germany, Denmark, Italy, Spain, and Brazil.</p> <p><code>OFF</code> turns CDN off. For example, the number 3,045,000.76 is represented as 3,045,000.76. This value is the default. OFF should be used for the USA, Canada, Mexico, and the United Kingdom.</p> <p><code>SPACE</code> sets the decimal point as a comma, and the thousands separator as a space. For example, the number 3,045,000.76 is represented as 3 045 000,76. SPACE should be used for France, Norway, Sweden, and Finland.</p> <p><code>QUOTE</code> sets the decimal point as a comma and the thousands separator as an apostrophe. For example, the number 3,045,000.76 is represented as 3'045'000,76. QUOTE should be used for Switzerland.</p>

Parameter:	CENT-ZERO
Description:	Determines whether decimal-only numbers display a leading zero. The setting of CDN determines whether a decimal point or comma is the decimal separator.
Syntax:	<code>SET CENT-ZERO = {ON OFF}</code> where: ON Displays fractions with a leading zero. The fraction is preceded by either a decimal point or comma depending on the CDN setting. OFF Does not display a leading zero. The fraction is preceded by either a decimal point or comma depending on the CDN setting. This value is the default.
Parameter:	COLUMNSCROLL
Description:	Enables you to scroll by column within the panels of a report provided that the report is wider than the screen width.
Syntax:	<code>SET COLUMNSCROLL = {ON OFF}</code> where: ON Enables column scrolling to the right and left by pressing the PF10 key and the PF11 key, respectively. To scroll up and down within the same column, use the PF7 key and the PF8 keys. OFF Disables column scrolling. This value is the default.
Parameter:	COMPUTE
Description:	Controls the compilation of calculations when a request is executed. When set to ON, calculations in a DEFINE or MODIFY COMPUTE command are compiled into machine code at request time. This code is used to perform calculations at run time.
Syntax:	<code>SET COMPUTE = {NEW OLD}</code> where: NEW Specifies the new, compiled logic. NEW is the default. OLD Forces all calculations into the old logic until the FOCUS session is over, or the SET COMPUTE command is reset.

Parameter: COUNTWIDTH

Description: Expands the default format of COUNT fields from a 5-byte integer to a 9-byte integer.

Syntax: SET {COUNTWIDTH|LISTWIDTH} = {ON|OFF}

where:

ON

Expands the default format of COUNT fields from a five-byte integer to a nine-byte integer.

OFF

Does not expand the default format of COUNT fields from a five-byte integer to a nine-byte integer. This value is the default.

Parameter: DATEDISPLAY

Description: Controls the display of a base date. Previously, TABLE always displayed a blank when a date read from a file matched the base date or a field with a smart date format had the value 0. The following shows the base date for each supported date format:

Format	Base Date
YMD and YYMD	1900/12/31
YM and YYM	1901/01
YQ and YYQ	1901/Q1
JUL and YYJUL	00/365 and 1900/365

Note: You cannot set DATEDISPLAY with the ON TABLE command.

Syntax: SET DATEDISPLAY = {ON|OFF}

where:

ON

Displays the base date if the data is the base date value.

OFF

Displays a blank if the date is the base date value. This value is the default.

Parameter:	<code>DATEFNS</code>
Description:	Loads the year 2000-compliant versions of the FUSELIB functions.
Syntax:	<code>SET DATEFNS = {<u>ON</u> OFF}</code> where: <code><u>ON</u></code> Loads the year 2000-compliant versions of the FUSELIB functions. This value is the default. <code>OFF</code> Uses non-year 2000-compliant functions.
Parameter:	<code>DATEFORMAT</code>
Description:	Specifies the order of the date components (month/day/year) when date-time values are entered in the formatted string and translated string formats. It makes a value's input format independent of the format of the variable to which it is being assigned.
Syntax:	<code>SET DATEFORMAT = <i>datefmt</i></code> where: <code><i>datefmt</i></code> Can be one of the following: MDY, DMY, YMD, or MYD. The U.S. English default format is MDY.
Parameter:	<code>DATETIME</code>
Description:	Sets time and date in reports. This command is useful for determining (statically or dynamically) exactly when your report was run. You can display the DATETIME value using any FOCUS date variable, for example, YMD, MDY, TOD, etc. If DATETIME is not set, the behavior of the FOCUS date variables remain the same.
Syntax:	<code>SET DATETIME = <i>option</i></code> where: <code><i>option</i></code> Is one of the following: <code><u>STARTUP</u></code> is the time and date when you began your FOCUS session. This setting is the default. <code><u>CURRENT</u> <u>NOW</u></code> changes each time it is interrogated. For example, if your batch job starts before midnight at 11:59 P.M., it won't complete until the next day. If DATETIME is set to <code><u>NOW</u> <u>CURRENT</u></code> , any reference to the variable gives the current date, not the date when the job started. <code><u>RESET</u></code> freezes the date and time of the current run for the rest of the session or until another SET DATETIME command is issued.

Parameter: DEFCEM

Description: Defines a default century globally or on a field-level for an application that does not contain an explicit century. DEFCEM is used in conjunction with YRTHRESH to interpret the current century according to the given values. When assigned globally, the time span created by these parameters applies to every 2-digit year used by the application unless you specify file-level or field-level values. (See YRTHRESH.)

Note: This same result can be achieved by including the FDEFCEM and FYRTHRESH attributes in the Master File.

Syntax: SET DEFCEM = {cc|19}

where:

cc

Is the default century. If you do not supply a value, cc defaults to 19, for the twentieth century.

Parameter: DISPLAY

Description: Is the PC display mode selection.

Syntax: SET DISPLAY = {OFF|PCCOLOR|PCMONO}

where:

option

Is one of the following:

OFF indicates no display mode is selected. This value is the default.

PCCOLOR indicates the display mode is color.

PCMONO indicates the display mode is black and white.

Parameter:	<code>DTSTRICT</code>
Description:	Controls how much error checking is done on date-time values when they are input by users, read from an alphanumeric transaction file, displayed, or used in functions.
Syntax:	<code>SET DTSTRICT = {<u>ON</u> OFF}</code> where: <u>ON</u> Invokes strict processing. This means that whenever a date-time value is input by a user, read from a transaction file, displayed, or returned by a function it is checked to make sure that the value represents a valid date and time. For example, a numeric month must be between 1 and 12, and the day must be within the number of days for the specified month. ON is the default value. If you attempt to enter a value that violates this rule, the following message displays: <code>(FOC177) INVALID DATE CONSTANT: dt_constant</code> <u>OFF</u> Does not invoke strict processing. Any date-time component can have any value within the constraint of the number of decimal digits allowed; for example, the month value can be 00 or 13 or 99, but not 115. Furthermore, the values do not have to be consistent; for example, any month in any year can have 30 or 31 days.
Parameter:	<code>EMPTYREPORT</code>
Description:	Controls the output generated when a TABLE request retrieves zero records. EMPTYREPORT is not supported with TABLEF. When a TABLEF request retrieves zero records, an empty report is always generated.
Syntax:	<code>SET EMPTYREPORT = {ON <u>OFF</u>}</code> where: <u>ON</u> Generates an empty report when zero records are found. <u>OFF</u> Does not generate an empty report when zero records are found. OFF is the default.

Parameter:**ERROROUT****Description:**

Controls how a batch FOCUS job step or Server responds to an error condition encountered in a procedure. This parameter cannot be set with the ON TABLE SET command.

When ERROROUT is set to ON, any error message generated terminates the job step and issues a return code of 8. Warning messages do not invoke this behavior. When ERROROUT is set to OFF, depending on the specific message, FOCUS determines whether FOCEXEC processing continues. Users can check a Dialogue Manager variable such as &FOCERRNUM and issue the following command to terminate FOCUS and set *n* as the return code:

```
-QUIT FOCUS n
```

On VM, if you include the QUEUE 'FIN' command in your batch FOCUS EXEC, and if FOCUS terminates as a result of the ERROROUT setting, the queued FIN command will cause CMS to issue a return code of -3, which will overwrite the ERROROUT return code. If you want to see the return code issued by Exit on Error, you can remove the QUEUE 'FIN' command from the EXEC and include the following command immediately after the 'EXEC FOCUS' command to exit and issue the return code:

```
exit rc
```

Note: The ERROROUT setting is ignored in an interactive session.

Syntax:

```
SET ERROROUT = {ON|OFF}
```

where:

ON

When an error message is generated in a batch FOCUS job step, sets the return code to 8 and terminates the job step. When an error message is generated on the Server, terminates the procedure and returns the error message to the client.

In addition, the following message displays to inform the user why the program terminated:

```
Exiting due to Exit on Error
```

OFF

Does not set a return code or automatically terminate a job step or procedure in response to any error message. This is the default value.

Parameter:

ESTRECORDS

Description:

Passes the estimated number of records to be sorted in the request. FOCUS queries using external sorts and including the parameter 'FILSZ=*En*' can diminish FOC909 errors. This parameter enables the sorting algorithms to estimate SORTWORK space requirements for each sort parameter request.

In order to make an accurate estimate for your ESTRECORDS setting, it is suggested that you run the report without an external sort in order to get a record count. If an attempt is made to SET ESTRECORDS from the FOCUS prompt, FOCPARM, or PROFILE FOCEXEC the following error is generated:

```
SET ESTRECORDS = n
```

```
(FOC36210) THE SPECIFIED PARAMETER CAN ONLY BE SET ON  
TABLE: ESTRECORDS
```

ESTRECORDS can only be set with the ON TABLE SET command within the TABLE, MATCH, or GRAPH request.

For CMS/SyncSort the 'FILSZ=*En*' parameter is ignored. Therefore, SET ESTRECORDS *n* has no effect.

Syntax:

```
SET ESTRECORDS = n
```

where:

n

Is the estimated number of records to be sorted.

Parameter:	EUROFILE
Description:	<p>Activates the data source that contains information for the currency you want to convert. This setting can be changed during a session to access a different currency data source. This parameter cannot be issued in a report request.</p> <p>Note: You cannot set any additional parameters on the same line as EUROFILE. FOCUS ignores any other parameters specified on the same line.</p>
Syntax:	<p><code>SET EUROFILE = {<i>ddname</i> OFF}</code></p> <p>where:</p> <p><i>ddname</i></p> <p>Is the name of the Master File for the currency data source you want to use. The <i>ddname</i> must refer to a read-only data source accessible by FOCUS. There is no default value.</p> <p>OFF</p> <p>Deactivates the current currency data source and removes it from memory.</p>
Parameter:	EXTAGGR
Description:	Uses external sorts to perform aggregation.
Syntax:	<p><code>SET EXTAGGR = {<u>ON</u> OFF NOFLOAT}</code></p> <p>where:</p> <p><u>ON</u></p> <p>Uses external sorts to perform aggregation.</p> <p>OFF</p> <p>Does not allow aggregation by an external sort.</p> <p>NOFLOAT</p> <p>Allows aggregation if there are no floating data fields present.</p>
Parameter:	EXTHOLD
Description:	Enables you to create HOLD files using an external sort.
Syntax:	<p><code>SET EXTHOLD = {ON <u>OFF</u>}</code></p> <p>where:</p> <p>ON</p> <p>Creates HOLD files using an external sort.</p> <p><u>OFF</u></p> <p>Does not create HOLD files using an external sort.</p>

Parameter: [EXTSORT](#)

Description: Calls an external sort for use with the TABLE, MATCH, and GRAPH commands.

If the report can be processed entirely in memory, external sorting does not occur. In order to determine if the report can be processed in memory, issue the ? STAT query after the TABLE, MATCH, or GRAPH command, and check the value of the SORT USED parameter.

Syntax: [SET](#) [EXTSORT](#) = {[ON](#)|[OFF](#)}

where:

[ON](#)

Enables FOCUS to pass records that are retrieved to an external sort. This value is the default.

[OFF](#)

Uses the FOCUS internal sorting procedure.

Parameter: [EXTTERM](#)

Description: Enables the use of extended terminal attributes.

Syntax: [SET](#) [EXTTERM](#) = {[ON](#)|[OFF](#)}

where:

[ON](#)

Enables the use of attributes. This value is the default.

[OFF](#)

Disables the use of attributes.

Parameter: [FIELDNAME](#)

Description: Controls the use of long field names (66 characters).

Syntax: [SET](#) [FIELDNAME](#) = {[NEW](#)|[NOTRUNC](#)|[OLD](#)}

where:

[NEW](#)

Supports long field names.

[NOTRUNC](#)

Does not support unique truncations.

[OLD](#)

Turns off support for long field names.

Parameter:	<code>FILE[NAME]</code>
Description:	Specifies a file to be used, by default, in commands. When you set a default file name, you can use that file without specifying its name.
Syntax:	<code>SET FILE[NAME] = <i>filename</i></code> where: <code><i>filename</i></code> Is a default file to be used in commands.
Parameter:	<code>FILTER</code>
Description:	Activates and deactivates filters. The SET FILTER command is limited to one line. To activate more filters to fit on one line repeat the SET FILTER command.
Syntax:	<code>SET FILTER = {<u>*</u> <i>filter</i> [<i>filter</i>]} IN <i>file</i> {ON <u>OFF</u>}</code> where: <u>*</u> Denotes all declared filters. This value is the default. ON Activates the filter. The maximum number of filters set ON for a file is limited by the number of IF/WHERE commands in these filters and should not exceed the standard FOCUS limit of IF/WHERE commands in any single TABLE request. <u>OFF</u> Deactivates the filter. This value is the default. <i>filter</i> Is the name of a filter as declared in the NAME = syntax of the FILTER FILE block.

Parameter: [FIXRETRIEVE](#)

Description: FOCUS HOLD files support keyed retrieval, which can greatly reduce the I/Os incurred in reading extract files. The performance gains are accomplished by using the SEGTYPE= parameter in the Master File to specify that the BY fields in the request be used as a logical key for sequential files. It allows you to stop the retrieval process when an equality test on this field holds true. This changes former behavior, as the interface previously read all of the records from the QSAM file and then passed them to FOCUS to apply the screening conditions when creating the final report.

Syntax: [SET](#) {[FIXRETRIVE](#)|[FIXF](#)} = {[ON](#)|[OFF](#)}

where:

[ON](#)

Stops the retrieval process when an equality test on this field holds true.

[OFF](#)

Does not stop the retrieval process when an equality test on this field holds true.

Parameter: [FOC144](#)

Description: Tells FOCUS to suppress warning message FOC144, which reads: “Warning: Testing in Independent sets of Data.”

Syntax: [SET](#) [FOC144](#) = {[NEW](#)|[OLD](#)}

where:

[NEW](#)

Displays the FOC144 warning message. This value is the default.

[OLD](#)

Suppresses the FOC144 warning message.

Parameter:	FOC2GIGDB
Description:	Enables two-gigabyte FOCUS data sources. Must be set in the FOCPARM profile. However, if the data source is in a FOCUS data source server, FOC2GIGDB must be set in HLIPROF.
Syntax:	<code>SET FOC2GIGDB = {ON OFF}</code> where: ON Enables support for FOCUS data sources larger than one-gigabyte. Note that an attempt to use FOCUS data sources larger than one-gigabyte in a release prior to FOCUS Version 7.1 can cause database corruption. OFF Disables support for FOCUS data sources larger than one-gigabyte. OFF is the default value.
Parameter:	FOCALLOC
Description:	This parameter applies only to MVS. Automatically allocates FOCUS files. Allocation is done based on Prefix.Master. FOCUS. The DISP will be SHR.
Syntax:	<code>SET {FOCALLOC FALLOC} = {ON OFF}</code> where: ON Automatically allocates FOCUS files. OFF Does not automatically allocate FOCUS files. This value is the default.
Parameter:	FOCSTACK
Description:	Is the amount of memory in thousands of bytes used by FOCSTACK, the stack of FOCUS commands from FOCEXECs awaiting execution. The maximum value of FOCSTACK depends on your current region size. You can also specify the parameter as FOCSTACK SIZE.
Syntax:	<code>SET FOCSTACK = {n 8}</code> where: n Is the amount of memory in thousands of bytes used by FOCSTACK. The default value is 8.

Parameter: `HDAY`

Description: Activates the holiday file that is used in conjunction with the data functions DATEDIF, DATEMOV, DATECVT, and DATEADD.

This setting is by default not set at all.

Syntax: `SET HDAY = string`

where:

string

Is the part of the name of the holiday file after HDAY. This string must be four characters long.

Parameter: `HIPERCACHE`

Description: Determines the default CACHE size in 4K pages when HiperFOCUS is activated. Can only be set in the FOCARM ERRORS profile.

Syntax: `SET HIPERCACHE = {cache|256}`

where:

cache

Is the default CACHE size in 4k pages when HiperFOCUS is activated. The default value is 256 (1M).

Parameter: `HIPEREXTENTS`

Description: Determines the permissible number of extents for HiperFOCUS (on MVS). Can only be set in the FOCARM ERRORS profile.

Syntax: `SET HIPEREXTENTS = {number|127}`

where:

number

Is the permissible number of extents. The default value is 127.

Parameter:	HIPERFILE
Description:	Is the maximum number of (4K) pages in an individual hiperspace. This is equivalent to the IBI Subsystem FILELIM parameter. If both are set, the lower is enforced. Can only be set in the FOCPARM ERRORS profile.
Syntax:	<code>SET HIPERFILE = {pages 524287}</code> where: <code>pages</code> Is the number of pages in an individual hiperspace. The default is 524287 (2GB).
Parameter:	HIPERFOCUS
Description:	Activates HiperFOCUS. If HiperFOCUS is not installed, this parameter is disabled.
Syntax:	<code>SET HIPERFOCUS = {ON OFF}</code> where: <code>ON</code> Activates HiperFOCUS. This value is the default. <code>OFF</code> Deactivates HiperFOCUS.
Parameter:	HIPERINSTALL
Description:	Installs or disables HiperFOCUS. Can only be set in the FOCPARM ERRORS profile.
Syntax:	<code>SET HIPERINSTSALL = {ON OFF}</code> where: <code>ON</code> Installs HiperFOCUS. <code>OFF</code> Disables HiperFOCUS. This value is the default.

Parameter:	HIPERLOCKED
Description:	Enables or disables processing of user interface commands such as SET HIPERFOCUS. Can only be set in the FOCPARM ERRORS profile.
Syntax:	<code>SET HIPERLOCKED = {ON OFF}</code> where: <code>ON</code> Disables processing of user interface commands. <code>OFF</code> Enables processing of user interface commands. This value is the default.
Parameter:	HIPERSPACE
Description:	Is the number of (4K) pages to aggregate for hiperspace. This is equivalent to the IBI Subsystem TCBLIM parameter. If both are set, the lower is enforced. Can only be set in the FOCPARM ERRORS profile.
Syntax:	<code>SET HIPERSPACE = {pages 524287}</code> where: <code>pages</code> Is the number of pages to aggregate for hiperspace. The default is 524287 (2GB).
Parameter:	HLISUTRACE
Description:	Used for debugging, records the last 20 events that the FOCUS Database Server (formerly called the sink machine) performed. The information is written to memory and is intended for use when reading a dump of the SU address space. This setting may only be set in the SU profile, HLIPROF.
Syntax:	<code>SET HLISUTRACE = {ON OFF}</code> where: <code>ON</code> Records the last 20 events that the FOCUS Database Server performed. This value is the default. <code>OFF</code> Does not record the last 20 events that the FOCUS Database Server performed.

Parameter: [HLISUDUMP](#)

Description: This setting is only used for debugging FOCUS Database Server problems and may only be set in the SU profile, HLIPROF.

Syntax: [SET HLISUDUMP = n](#)

where:

[n](#)

When set to 99999, a dump of the FOCUS Database Server address space will occur for any error on the server. The user abend code is set to 275. The user code will also be set to the error number.

Parameter: [HOLDATTR\[S\]](#)

Description: Includes the TITLE and ACCEPT attributes from the original Master File in the HOLD Master File. This setting does not affect the way fields are named in the HOLD Master File.

Syntax: [SET HOLDATTR = {ON|OFF|\[FOCUS\]\(#\)}](#)

where:

[ON](#)

Includes the TITLE attribute from the original Master File in HOLD Master Files for HOLD files of any format. The ACCEPT attribute is included in the HOLD Master File when the HOLD file is in FOCUS format.

[OFF](#)

Does not include the TITLE or ACCEPT attributes from the original Master File in the HOLD Master File.

[FOCUS](#)

Includes the TITLE and ACCEPT attributes in HOLD Master Files when the HOLD file is in FOCUS format. This value is the default.

Parameter:	HOLDLIST
Description:	Determines what fields in a report request are included in the HOLD file.
Syntax:	<pre>SET HOLDLIST = {PRINTONLY ALL}</pre> <p>where:</p> <p>PRINTONLY</p> <p>Includes only those fields in the HOLD file that are displayed by the report request.</p> <p>ALL</p> <p>Includes all fields referenced in a request in the HOLD file, including both computed fields and fields referenced in a COMPUTE command. This value is the default. (OLD may be used as a synonym for ALL.)</p>
Parameter:	HOLDSTAT
Description:	Includes comments and DBA information in HOLD Master Files. This information can be from the HOLDSTAT ERRORS file supplied by Information Builders, or a user-specified file.
Syntax:	<pre>SET HOLDSTAT = {ON OFF <i>name</i>}</pre> <p>where:</p> <p>ON</p> <p>Derives comments and DBA information from the holdstat.mas or errors.mas file in UNIX and NT. In MVS, this information is derived from the member HOLDSTAT in the PDS allocated to the ddname MASTER or ERRORS. In CMS, it is derived from the file HOLDSTAT MASTER or HOLDSTAT ERRORS.</p> <p>OFF</p> <p>Does not include information from the HOLDSTAT file in the HOLD Master File. This value is the default.</p> <p><i>name</i></p> <p>Specifies a HOLDSTAT file, created by the end user, whose information is included in the HOLD Master File.</p>

Parameter:	HOTMENU
Description:	Automatically displays the Hot Screen PF key legend at the bottom of the Hot Screen report.
Syntax:	SET HOTMENU = { ON OFF }
	where:
	ON Displays the PF key legend.
	OFF Does not display the PF key legend. To see the PF key legend, the user must press PF1. OFF is the default.
Parameter:	IBMLE
Description:	Controls the LE run-time environment by identifying which LE libraries to load
Syntax:	SET IBMLE = { OFF ON ALL }
	where:
	OFF Loads the libraries for LE-compiled C and COBOL subroutines. OFF is the default value.
	ON Adds the libraries for LE-compiled PL/I subroutines to the C and COBOL libraries. Once the ON setting has been established, you cannot issue the OFF setting.
	ALL Adds the libraries for LE-compliant FORTRAN and PL/I subroutines (if they are not already loaded) to the C and COBOL libraries. Once the ALL setting has been established, you cannot issue the OFF or ON setting.
Parameter:	IMMEDTYPE
Description:	Used with TOE, tells FOCUS where to send line mode output.
Syntax:	SET IMMEDIATE = { ON OFF }
	where:
	ON Sends all line mode output, such as -TYPE, to the Output Window as it is executed, line by line.
	OFF Buffers all line mode output. The output appears in the Output Window as a new full screen. This value is the default.

Parameter:	INDEX
Description:	The indexing scheme used for indexes (fields specified with FIELDTYPE=I keywords in the Master Files).
Syntax:	SET INDEX [TYPE] = {NEW OLD} where: NEW Creates a binary tree index. This value is the default. OLD Creates a hash index.
Parameter:	JOINOPT
Description:	Allows the joining of two files that contain different numeric data types.
Syntax:	SET JOINOPT = {NEW OLD} where: NEW Allows the joining of files that contain different numeric data types. OLD Does not allow the joining of files that contain different numeric data types.
Parameter:	KEEPDEFINES
Description:	Controls whether a virtual field created for a host or joined structure is retained after a JOIN command is run. This parameter applies when a DEFINE command precedes the JOIN command.
Syntax:	SET KEEPDEFINES = {ON OFF} where: ON Retains the virtual field after a JOIN command is run. OFF Clears the virtual field after a JOIN command is run. This value is the default.

Parameter: `LANG[UAGE]`
Description: Specifies the National Language Support (NLS) environment.
Syntax: `SET LANG[UAGE] = value`

where:

`value`

Is a language from the following list. The ID, name, or abbreviation can be used to specify the language.

The default value for TERM is IBM3270, which does not support DBCS.

ID	Name	Abbreviation
1	ENGLISH	AME
1	AMENGLISH	AME
86	T-CHINESE	ROC
45	DANISH	DAN
31	DUTCH	DUT
358	FINNISH	FIN
33	FRENCH	FRE
49	GERMAN	GER
30	GREEK	GRE
972	HEBREW	HEB
39	ITALIAN	ITA
81	JAPANESE	JPN
47	NORWEGIAN	NOR
351	PORTUGUESE	POR
34	SPANISH	SPA
46	SWEDISH	SWE
90	TURKISH	TUR
44	UKENGLISH	UKE

Parameter:	LEADZERO
Description:	Leading zeros are truncated in Dialogue Manager strings. The functions in FOCUS, when called in Dialogue Manager, may return a numeric result. If the format of the result is YMD and contains a 00 for the year, the 00 is truncated.
Syntax:	<code>SET LEADZERO = {ON OFF}</code> where: <code>ON</code> Allows the display of leading zeros if they are present. <code>OFF</code> Truncates leading zeros if they are present.
Parameter:	LEFTMARGIN
Description:	This parameter applies only to StyleSheets. Sets the left boundary for report contents on a page.
Syntax:	<code>SET LEFTMARGIN = {value .250}</code> where: <code>value</code> Is the left boundary of report contents on a page. The default is .25 inches.
Parameter:	LINES
Description:	Sets the maximum number of lines of printed output that appear on a page, from the heading at the top to the footing on the bottom. If this value is less than the value set for PAPER, the difference provides a bottom margin. FOCUS never puts more lines on a page than the LINES parameter specifies, but may put less. The value of LINES can range between 1 and 999999; specify 999999 for continuous forms. Note: When you use SKIP-LINE in a report, always set LINES to at least one less than the value for PAPER. This avoids unintentional page beaks at the bottom of the page. When the STYLESHEET parameter is in effect, the setting for LINES is ignored.
Syntax:	<code>SET LINES = {n 57}</code> where: <code>n</code> Is the maximum number of lines of printed output that appear on a page. The default value is 57.

Parameter:	MASTER
Description:	<p>This parameter applies only to the FUSION option.</p> <p>New Master Files pass for blank delimited Master Files, which use the new FUSION syntax.</p>
Syntax:	<p>SET MASTER = {NEW OLD}</p> <p>where:</p> <p>NEW</p> <p>Enables use of blank delimited Master File syntax (FUSION syntax) in addition to comma-delimited syntax.</p> <p>OLD</p> <p>Accepts only comma-delimited Master File syntax. This value is the default.</p>
Parameter:	MAXLRECL
Description:	<p>Defines the maximum record length for an external file with OCCURS segments. The default is 0. However, FOCUS can read a 16K recl by default. This may be set to a maximum of 32K.</p>
Syntax:	<p>SET MAXLRECL = {n 0}</p> <p>where:</p> <p>n</p> <p>Is the maximum record length for an external file with OCCURS segments. The default value is 0.</p>
Parameter:	MESSAGE
Description:	<p>Controls the display of informational messages.</p>
Syntax:	<p>SET {MESSAGE MSG} = {ON OFF}</p> <p>where:</p> <p>ON</p> <p>Displays informational messages. This value is the default.</p> <p>OFF</p> <p>Suppresses both informational messages and carets that appear when FOCUS executes commands in procedures. FOCUS still displays error messages, and the carets that prompt for input.</p>

Parameter: [MINIO](#)

Description: This parameter applies only to MVS.

Improves performance by reducing I/O operations up to fifty percent when accessing FOCUS data sources under MVS. This is a buffering technique.

With FOCUS data sources that are not disorganized, MINIO can greatly reduce the number of I/O operations for TABLE and MODIFY commands. The actual I/O reduction will vary depending on data source structure and average number of children segments per parent segment. By reducing I/O operations, elapsed time for TABLE and MODIFY commands also drop.

Syntax: [SET MINIO](#) = {[ON](#)|[OFF](#)}

where:

[ON](#)

Does not read a block more than once; the number of reads performed will be the same as the number of tracks present. This results in an overall reduction in elapsed times when reading and writing. This value is the default.

[OFF](#)

Disables MINIO.

Parameter:	MULTIPATH
Description:	Controls testing on independent paths.
Syntax:	<pre>SET MULTIPATH = {SIMPLE COMPOUND}</pre> <p>where:</p> <p>SIMPLE</p> <p>Includes a parent segment in the report output if:</p> <ul style="list-style-type: none">• It has at least one child that passes its screening conditions.• It lacks any referenced child on a path, but the child is optional (see the <i>Creating Reports</i> manual). <p>SIMPLE is the default value for FOCUS for S/390.</p> <p>The (FOC144) warning message is generated when a request screens data in a multi-path report.</p> <pre>(FOC144) WARNING. TESTING IN INDEPENDENT SETS OF DATA:</pre> <p>COMPOUND</p> <p>Includes a parent in the report output if it has <i>all</i> of its required children (see the <i>Creating Reports</i> manual). The COMPOUND setting does not generate the (FOC144) warning message.</p> <p>COMPOUND is the default value for iWay and WebFOCUS.</p> <p>The segment rule is applied level by level as FOCUS descends the data source/view hierarchy. That is, a parent segment's existence depends on the child segment's existence and the child segment depends on the grandchild's existence, and so on for the full data source tree.</p>
Parameter:	NODATA
Description:	Determines the character string that indicates missing data in a report. The NODATA parameter can be abbreviated to NA.
Syntax:	<pre>SET {NODATA NA} = {string .}</pre> <p>where:</p> <p>string</p> <p>Is the character string that indicates missing data in reports. The default is a period.</p>

Parameter:	<code>ONLINE-FMT</code>
Description:	Determines the format of report output. StyleSheet reports are generated in PostScript format. Styled reports can only be printed on a PostScript printer.
Syntax:	<code>SET ONLINE-FMT = {STANDARD POSTSCRIPT}</code> where: STANDARD Produces the report as un-styled character-based output. This value is the default. POSTSCRIPT Saves the report output to a PostScript file with the name PSOUT. In MVS, the PostScript formatted report output is in a variable length PDS allocated to the ddname PS. In CMS, the output is in a file with the file type PS. The parameters set with the SET STYLESHEET command are in effect. PS can be used as a synonym for POSTSCRIPT.
Parameter:	<code>ORIENTATION</code>
Description:	This parameter applies to StyleSheets. Specifies the page orientation for styled reports.
Syntax:	<code>SET ORIENTATION = {PORTRAIT LANDSCAPE}</code> where: PORTRAIT Displays the page in portrait style. This value is the default. LANDSCAPE Displays the page in landscape style.

Parameter: `PAGE[-NUM]`

Description: Controls the numbering of output pages.

Syntax: `SET PAGE[-NUM] = option`
`option`

Is one of the following:

`ON` displays the page number on the upper left-hand corner of the page. This value is the default.

`OFF` suppresses page numbering.

`NOPAGE` suppresses page breaks, causing the report to be printed as a continuous page. When `PAGE` is set to `NOPAGE`, the `LINES` parameter controls where column headings are printed.

`TOP` omits the line at the top of each page of the report output for the page number and the blank line that follows it. The first line of report output contains the heading, if one was specified, or the column titles if there is no heading.

Note: The settings `ON`, `TOP`, and `OFF` include the carriage control character 1 in the first column of each page.

Parameter: `PAGESIZE`

Description: Specifies the page size for StyleSheets. For optimal report appearance, the actual paper size must match your setting for `PAGESIZE`. If it does not, your report or your report will be cropped or contain extra blank spaces.

Syntax: `SET PAGESIZE = size`

where:

`size`

Specifies the page size. If the actual paper size does not match the `PAGESIZE` setting, your report will either be cropped or contain extra blank space. The options are:

`LETTER` sets the page size to 8.5 x 11 inches.

`ENVELOPE-PERSONAL` sets the page size to 3.625 x 6.5 inches.

`ENVELOPE-MONARCH` sets the page size to 3.875 x 7.5 inches.

`ENVELOPE-9` sets the page size to 3.875 x 8.875 inches.

`ENVELOPE-10` sets the page size to 4.125 x 9.5 inches.

`ENVELOPE-11` sets the page size to 4.5 x 10.375 inches.

`ENVELOPE-12` sets the page size to 4.5 x 11 inches.

`ENVELOPE-14` sets the page size to 5 x 11.5 inches.

`STATEMENT` sets the page size to 5.5 x 8.5 inches.

`EXECUTIVE` sets the page size to 7.5 x 10.5 inches.

`GERMAN-STANDARD-FANFOLD` sets the page size to 8.5 x 12 inches.

`GERMAN-LEGAL-FANFOLD` sets the page size to 8.5 x 13 inches.

`FOLIO` sets the page size to 8.5 x 13 inches.

`LEGAL` sets the page size to 8.5 x 14 inches.

`10X14` sets the page size to 10 x 14 inches.

`TABLOID` sets the page size to 11 x 17 inches.

`C` sets the page size to 17 x 22 inches.

`D` sets the page size to 22 x 34 inches.

`E` sets the page size to 34 x 44 inches.

`US-STANDARD-FANFOLD` sets the page size to 14.875 x 11 inches.

`LEDGER` sets the page size to 17 x 11 inches.

`ENVELOPE-DL` sets the page size to 4.3 x 8.6 inches.

`ENVELOPE-ITALY` sets the page size to 4.3 x 9.1 inches.

`ENVELOPE-C6` sets the page size to 4.5 x 6.375 inches.

`ENVELOPE-C65` sets the page size to 4.5 x 9 inches.

`A5` sets the page size to 5.8 x 8.25 inches.

`ENVELOPE-C5` sets the page size to 6.4 x 9 inches.

`ENVELOPE-B5` sets the page size to 6.9 x 9.8 inches.

`ENVELOPE-B6` sets the page size to 6.9 x 4.9 inches.

`B5` sets the page size to 7.2 x 10.1 inches.

`A4` sets the page size to 8.25 x 11.7 inches.

`QUARTO` sets the page size to 8.5 x 10.8 inches.

`ENVELOPE-C4` sets the page size to 9 x 12.75 inches.

`ENVELOPE-B4` sets the page size to 9.8 x 13.9 inches.

`B4` sets the page size to 9.8 x 13.9 inches.

`A3` sets the page size to 11.7 x 16.8 inches.

`ENVELOPE-C3` sets the page size to 12.75 x 18 inches.

Parameter:

PANEL

Description:

Sets the maximum line width, in characters, of a report panel for a screen or printer. If report output exceeds this value, the output is partitioned into several panels. For example, if you set PANEL to 80, the first 80 characters of a record appear on the first panel, the second 80 characters appear on the second panel, and so on.

When printing a report to your screen, the ideal value for the PANEL parameter is the width of your screen (usually 80). When printing to your printer, the ideal value for PANEL is the print width of your printer (usually 132). If PANEL is larger or set to 0, long report lines wrap around the screen or page.

When the BYPANEL parameter is OFF, a report can be divided into a maximum of four panels. If SET BYPANEL has a value other than OFF, the report may be divided into 99 panels.

When the STYLESHEET parameter is in effect, PANEL is ignored.

Syntax:

```
SET PANEL = {0|n}
```

where:

n

Is the maximum line width, in characters, of a report panel.

0

Does not divide the report into panels. Long report lines wrap around the screen or page. This value is the default.

Parameter: `PAPER`

Description: Specifies the physical length of the paper, in lines, for printed output. You derive this value by multiplying the length of the paper, in inches, by the number of lines printed per inch. For example, if your printer prints six lines per inch on standard 11 inch forms, PAPER should be set to 66. If you are placing a footing at the bottom of the page, this value should be less; in this case, 62. Valid values for PAPER are numbers between 1 and 999999. Specify 999999 for continuous forms.

Note: When the STYLESHEET parameter is in effect, the setting for PAPER is ignored.

Syntax: `SET PAPER = {n|66}`

where:

n

Is the length of paper, in lines, for printed output. Valid values are numbers between 1 and 999999. The value 999999 denotes the use of continuous forms. The default value is 66.

Parameter: `PASS`

Description: Enables user access to a data source or stored procedure protected by Information Builders security.

Syntax: `SET PASS = password [IN filename]`

where:

password

Is the password that allows access to data sources protected by Information Builders database security.

filename

Is the FOCUS data source or stored procedure protected by security.

Parameter: [PAUSE](#)

Description: Pauses before displaying a FOCUS report on the terminal. When you use a printing terminal, this parameter allows you to adjust the paper before printing the report.

When the SCREEN parameter is ON, the PAUSE parameter is set ON (until you set the PAUSE parameter to OFF). If you set the SCREEN parameter to OFF, the PAUSE parameter is set to OFF. Note that you can change the PAUSE parameter without affecting the SCREEN parameter.

This setting does not affect offline printing (routing output to a system printer).

Syntax: [SET PAUSE = {ON|OFF}](#)

where:

[ON](#)

Pauses before displaying a FOCUS report. This value is the default.

[OFF](#)

Does not pause before displaying a FOCUS report.

Parameter: [PCOMMA](#)

Description: Enables the retrieval of comma-delimited files created by a PC application or the HOLD FORMAT COM command.

By default, when a Master File specifies SUFFIX=COM, incoming alphanumeric values are not enclosed in double quotation marks, and each record is terminated with a comma and dollar sign (,\$) character combination. This format does not support retrieval of most comma-delimited files produced by a PC application.

Syntax: [SET PCOMMA = {ON|OFF}](#)

where:

[ON](#)

Enables the retrieval of comma-delimited data sources created by a PC application. It indicates that alphanumeric data is enclosed in double quotation marks and each record is completely contained on one line and is terminated with a carriage return and line feed.

[OFF](#)

Does not enable the retrieval of comma-delimited data sources created by a PC application. It indicates that alphanumeric data is not enclosed in double quotation marks and each record is terminated with a comma and dollar sign. This value is the default.

Parameter:	<code>PFnn</code>
Description:	<p>Assigns a function to the PF key specified by <i>nn</i>, enabling you to change the current PF key setting when using FIDEL (and also, under certain conditions, within the Window facility).</p> <p>The current settings are displayed by the ? PFKEY command.</p>
Syntax:	<p><code>SET PFnn = function</code></p> <p>where:</p> <p><code>nn</code></p> <p>Is the PF key you are assigning a function to.</p> <p><code>function</code></p> <p>Is the function to assign to the PF key specified by PFnn.</p>
Parameter:	<code>PREFIX</code>
Description:	<p>This parameter applies only to MVS.</p> <p>Specifies the prefix of existing data sets automatically allocated by FOCUS.</p>
Syntax:	<p><code>SET PREFIX = prefix</code></p> <p>where:</p> <p><code>prefix</code></p> <p>Specifies of the prefix of existing data sets automatically allocated by FOCUS. The default setting in TSO is your user ID; the default setting in batch is FOCUS.</p>
Parameter:	<code>PRINT</code>
Description:	<p>Specifies the report output destination.</p> <p>You can enter ONLINE and OFFLINE as separate commands that have the same effect as specifying ONLINE and OFFLINE as PRINT settings.</p>
Syntax:	<p><code>SET PRINT = {<u>ONLINE</u> OFFLINE}</code></p> <p>where:</p> <p><u><code>ONLINE</code></u></p> <p>Prints report output to the terminal.</p> <p><code>OFFLINE</code></p> <p>Prints report output to the system printer.</p>

Parameter:	PRINTPLUS
Description:	<p>Introduces enhancements to the display alternatives offered by the FOCUS Report Writer. To force a break at a specific spot, you must use NOSPLIT.</p> <p>PRINTPLUS is not supported with StyleSheets. Problems may be encountered if HOTSCREEN is set to OFFLINE.</p>
Syntax:	<p>SET {PRINTPLUS PRTPLUS} = {<u>ON</u> OFF}</p> <p>where:</p> <p><u>ON</u></p> <p>Handles the PAGE-BREAK internally to provide the correct spacing of pages, NOSPLIT is handled internally and you can perform RECAPs in cases where pre-specified conditions are met. Additionally, a Report SUBFOOT now prints above the footing instead of below it. ON is the default.</p> <p>OFF</p> <p>Does not support StyleSheets.</p>

Parameter:	QUALCHAR																		
Description:	Specifies the qualifying character to be used in qualified field names.																		
Syntax:	SET QUALCHAR = {character .}																		
	where:																		
	<i>character</i>																		
	Is a valid qualifying character. They include:																		
	<table><tr><td>.</td><td>period</td><td>(hex 4B)</td></tr><tr><td>:</td><td>colon</td><td>(hex 7A)</td></tr><tr><td>!</td><td>exclamation point</td><td>(hex 5A)</td></tr><tr><td>%</td><td>percent sign</td><td>(hex 6C)</td></tr><tr><td> </td><td>broken vertical bar</td><td>(hex 6A)</td></tr><tr><td>\</td><td>backslash</td><td>(hex E0)</td></tr></table>	.	period	(hex 4B)	:	colon	(hex 7A)	!	exclamation point	(hex 5A)	%	percent sign	(hex 6C)		broken vertical bar	(hex 6A)	\	backslash	(hex E0)
.	period	(hex 4B)																	
:	colon	(hex 7A)																	
!	exclamation point	(hex 5A)																	
%	percent sign	(hex 6C)																	
	broken vertical bar	(hex 6A)																	
\	backslash	(hex E0)																	
	The period is the default character. The use of the other qualifying characters listed above is restricted; they should not be used with 66-character field names.																		
	If the qualifying character is a period, you can use any of the other characters listed above as part of a field name. If you change the default qualifying character to a character other than the period, then you cannot use that character in a field name.																		
	In VM, if the TERM tabchar is ON or if the CMS INPUT command includes the broken vertical bar (hex 6A), then the broken vertical bar cannot be the qualifying character. To query INPUT, type Q INPUT at the CMS prompt.																		
Parameter:	QUALTITLES																		
Description:	Uses qualified column titles in report output when duplicate field names exist in a Master File. A qualified column title distinguishes between identical field names by including the segment name.																		
Syntax:	SET QUALTITLES = {ON OFF}																		
	where:																		
	ON																		
	Uses qualified column titles when duplicate field names exist and FIELDNAME is set to NEW.																		
	OFF																		
	Disables qualified column titles. This value is the default.																		

Parameter:	REBUILDMSG
Description:	Allows for direct control over the frequency with which REBUILD issues messages.
Syntax:	SET {REBUILDMSG REMSG} = <i>n</i> where: <i>n</i> Is any number.
Parameter:	RECAP-COUNT
Description:	Includes lines containing a value created with RECAP when counting the number of lines per page for printed output. The number of lines per page is determined by the LINES parameter.
Syntax:	SET RECAP-COUNT = {ON OFF} where: ON Counts lines containing a value created with RECAP. OFF Does not count lines containing a value created with RECAP. This value is the default.
Parameter:	RECORDLIMIT
Description:	Limits the number of records retrieved.
Syntax:	SET RECORDLIMIT = { <i>n</i> RECORDLIMIT} where: <i>n</i> Is the maximum number of records to be retrieved. RECORDLIMIT Respects explicit RECORDLIMIT values only.

Parameter: [RIGHTMARGIN](#)

Description: This parameter applies to StyleSheets.
Sets the right boundary for report contents on a page.

Syntax: `SET RIGHTMARGIN = {value|.25}`

where:

[value](#)

Is the right boundary of report contents on a page. The default value is .25 inches.

Parameter: [SAVEMATRIX](#)

Description: Preserves the internal matrix and keeps it available for subsequent RETYPE, HOLD, SAVE, SAVB, and REPLOTT commands when followed by Dialogue Manager commands.

Syntax: `SET SAVEMATRIX = {ON|OFF}`

where:

[ON](#)

Saves the last internal matrix generated.

[OFF](#)

Does not guarantee that the internal matrix will be available.
This value is the default.

Parameter: [SBORDER](#)

Description: Generates a solid border on the screen for full-screen mode.

If the screen appears to be generated incorrectly, it is possible that the terminal does not support this new feature; change the setting to OFF to correct the situation.

The amper variable &FOCSBORDER contains the value of the SBORDER setting. &FOCSBORDER may be included in the Dialogue Manager -TYPE command.

Syntax: `SET SBORDER = {ON|OFF}`

where:

[ON](#)

Enables solid borders. This value is the default.

[OFF](#)

Enables dashed (nonsolid) borders.

Parameter:	SCREEN
Description:	<p>Selects the Hot Screen facility.</p> <p>When the SCREEN parameter is ON, the PAUSE parameter is set ON (until you set the PAUSE parameter OFF). If you set the SCREEN parameter OFF, the PAUSE parameter is set OFF. Note that you can change the PAUSE parameter without affecting the SCREEN parameter.</p>
Syntax:	<p><code>SET SCREEN = {ON OFF PAPER}</code></p> <p>where:</p> <p>ON</p> <p>Activates the Hot Screen facility. This value is the default.</p> <p>OFF</p> <p>Deactivates the Hot Screen facility.</p> <p>PAPER</p> <p>Activates the Hot Screen facility and causes FOCUS to use the settings for LINES and PAPER parameters to format screen display.</p>
Parameter:	SHADOW
Description:	Activates the Absolute File Integrity feature.
Syntax:	<p><code>SET SHADOW [PAGE] = {ON OFF OLD}</code></p> <p>where:</p> <p>ON</p> <p>Activates the Absolute File Integrity feature. The maximum number of pages shadowed is 256K.</p> <p>OFF</p> <p>Deactivates the Absolute File Integrity feature. OFF is the default.</p> <p>OLD</p> <p>Indicates that your FOCUS file was created before Version 7.0. This means that the maximum number of pages shadowed is 63,551.</p>

Parameter:	<code>SHIFT</code>
Description:	Controls the use of “shift” strings.
Syntax:	<code>SET SHIFT = {ON OFF}</code> where: <code>ON</code> Specifies a shift string for Hebrew or DBCS (double-byte character support). <code>OFF</code> Indicates that <code>SHIFT</code> is not in effect. <code>OFF</code> is the default.
Parameter:	<code>SORTLIB</code>
Description:	This parameter applies only to FOCUS VM/CMS. Tells FOCUS which sort package is installed at your site.
Syntax:	<code>SET SORTLIB = <i>option</i></code> where: <code><i>option</i></code> Is one of the following: <code>VMSORT</code> is the VMSORT sort package. <code>SYNCSORT</code> is the SYNCSORT sort package. <code>DFSORT</code> is the DFSORT sort package. <code>SITEDEFINED</code> is used if not <code>VMSORT</code> , <code>SYNCSORT</code> , or <code>DFSORT</code> . This sort package must be installed in <code>SORTLIB</code> <code>TXTLIB</code> in order for FOCUS to find it.
Parameter:	<code>SPACES</code>
Description:	Sets the number of spaces between columns in a report.
Syntax:	<code>SET SPACES = {AUTO <i>n</i>}</code> where: <code>AUTO</code> Automatically places either one or two spaces between columns. This value is the default. <code><i>n</i></code> Is the number of spaces to place between columns of a report. Valid values are integers between 1 and 8.

Parameter:	<code>SQLTOPTTF</code>
Description:	Enables the SQL Translator to generate TABLEF commands instead of TABLE commands.
Syntax:	<code>SET SQLTOPTTF = {ON OFF}</code> where: ON Generates TABLEF commands when possible. For example, a TABLEREF command is generated if there is no JOIN or GROUP BY command. This value is the default. OFF Always generates TABLE commands.
Parameter:	<code>SQUEEZE</code>
Description:	This parameter applies only to the StyleSheet feature. Determines the column width in report output. The column width is based on the size of the data value or column title, or on the field format defined in the Master File.
Syntax:	<code>SET SQUEEZE = {ON OFF}</code> where: ON Assigns column widths based on the widest data value or widest column title, whichever is longer. This value is the default. OFF Assigns column widths based on the field format specified in the Master File. This value pads the column width to the length of the column title or field format descriptions, whichever is greater.

Parameter:	<code>STYLE[SHEET]</code>
Description:	Controls the format of report output by accepting or rejecting StyleSheet parameters. These parameters specify formatting options such as page size, orientation, and margins.
Syntax:	<code>SET STYLE[SHEET] = {stylesheet ON OFF}</code> where: <code>stylesheet</code> Is the name of the StyleSheet file. For UNIX and NT, this is the name of the StyleSheet file without the file extension .sty. For MVS, this is the member name in the PDS allocated to ddname FOCSTYLE by a DYNAM command. For CMS, this is the name of a file with file type FOCSTYLE. For a PDF or PostScript report, FOCUS uses the page layout settings for UNITS, TOPMARGIN, BOTTOMMARGIN, LEFTMARGIN, RIGHTMARGIN, PAGESIZE, ORIENTATION, and SQUEEZE; the settings for LINES, PAPER, PANEL, and WIDTH are ignored. ON FOCUS uses the page layout settings for UNITS, TOPMARGIN, BOTTOMMARGIN, LEFTMARGIN, RIGHTMARGIN, PAGESIZE, ORIENTATION, and SQUEEZE; the settings for LINES, PAPER, PANEL, and WIDTH are ignored. OFF FOCUS uses the settings for LINES, PAPER, PANEL, and WIDTH; the settings for UNITS, TOPMARGIN, BOTTOMMARGIN, LEFTMARGIN, RIGHTMARGIN, PAGESIZE, ORIENTATION, and SQUEEZE are ignored.
Parameter:	<code>SUMPREFIX</code>
Description:	When an external sort product performs aggregation of alphanumeric or smart date formats, the order of the answer set returned differs from the order of the FOCUS sorted answer sets.
Syntax:	<code>SET SUMPREFIX = {FST LST}</code> where: FST Displays the first value in cases of data aggregation of alphanumeric or smart date data types. LST Displays the last value in cases of data aggregation of alphanumeric or smart date data types.

Parameter:	<code>SUSI</code>
Description:	See <i>Simultaneous Usage for OS/390 and MVS</i> .
Parameter:	<code>SUTABSIZE</code>
Description:	See <i>Simultaneous Usage for OS/390 and MVS</i> .
Parameter:	<code>TEMP[DISK]</code>
Description:	<p>This parameter applies only to CMS.</p> <p>Determines the disk FOCUS uses for temporary work space, and to store extract files (HOLD and SAVE).</p>
Syntax:	<p><code>SET TEMP[DISK] = disk</code></p> <p>where:</p> <p><code>disk</code></p> <p>Is the disk FOCUS uses for temporary workspace, and to store extract files.</p>
Parameter:	<code>TERM</code>
Description:	Selects the terminal type.
Syntax:	<p><code>SET TERM[INAL] = {type IBM3270}</code></p> <p>where:</p> <p><code>type</code></p> <p>Is the terminal type. The options are:</p> <p>IBM3270 is the default value. It does not support DBCS.</p> <p>IBM5550 specifies an IBM 5550 or a PS/55 terminal. Supports DBCS.</p> <p>F6650 specifies a Facom F-6650 terminal. Supports DBCS.</p> <p>H56020 specifies a Hitachi H-560/20 terminal. Supports DBCS.</p>

Parameter: [TESTDATE](#)

Description: Temporarily alters the system date in order to test a dynamic window. That is, it allows you to simulate clock settings beyond the year 1999 to determine the behavior of your program.

Only use TESTDATE for testing purposes with test data. The value of TESTDATE affects all reserved variables that retrieve the current date from the system. Setting TESTDATE also affects anywhere in FOCUS that a date is used (such as CREATE, MODIFY, MAINTAIN) but does not affect the date referenced directly from the system.

TESTDATE can either be equal to TODAY or a date in the format YYYYMMDD. If anything else is entered the following message is displayed:

[TESTDATE MUST BE YYYYMMDD OR TODAY](#)

Syntax: [SET TESTDATE = {yyyymmdd|TODAY}](#)

where:

[yyyymmdd](#)

Is an 8-digit date in the format YYYYMMDD.

[TODAY](#)

Is the current date. This value is the default.

Parameter: [TEXTFIELD](#)

Description: Preserves downward compatibility with prior FOCUS releases. FOCUS text fields were enhanced significantly with Version 7.0.

Note: FOCUS Version 7.0 and higher preserves text fields exactly as they are entered into a data source with the ON MATCH/NOMATCH TED command. See the *Overview and Operating Environments* manual for additional information.

Syntax: [SET {TEXTFIELD|TXTFIELD} = {\[OLD\]\(#\)|\[NEW\]\(#\)}](#)

where:

[OLD](#)

Enables you to use text field data in prior releases of FOCUS when that data has been created or modified in Version 7.0. This value is the default.

[NEW](#)

Disables the ability to use text field data in prior FOCUS releases when that data has been created or modified in Version 7.0.

Parameter:	<code>TITLE</code>
Description:	Uses pre-defined column titles in the Master File as column titles in report output.
Syntax:	<code>SET TITLE[S] = {<u>ON</u> OFF}</code> where: <u>ON</u> Uses pre-defined column titles in the Master File as column titles in report output. This value is the default. <u>OFF</u> Uses the field names in the Master File as column titles in report output.
Parameter:	<code>TOPMARGIN</code>
Description:	This parameter applies to StyleSheets. Sets the top boundary on a page for report output.
Syntax:	<code>SET TOPMARGIN = {value .25}</code> where: value Is the top boundary on a page for report output. The default value is .25 inches.
Parameter:	<code>TRACKIO</code>
Description:	MVS FOCUS gathers more pages to fill a track before reading or writing the pages to disk. This results in significant reductions in I/O requirements and in elapsed time for FOCUS files.
Syntax:	<code>SET TRACKIO = {<u>ON</u> OFF}</code> where: <u>ON</u> Enables FOCUS to fill a track before reading or writing to disk. This value is the default. <u>OFF</u> Does not fill a track before reading and writing to a disk.

Parameter: [TRANTERM](#)

Description: Displays extended currency symbols on TSO. By default, when displaying report output in TSO without HotScreen (SET SCREEN=OFF), the extended currency symbols do not display because the terminal I/O procedures translate all terminal output to characters that appear in USA EBCDIC keyboard layouts and code charts.

Syntax: [SET TRANTERM](#) = {[ON](#)|[OFF](#)}

where:

[ON](#)

Does not display extended currency symbols. This value is the default.

[OFF](#)

Displays extended currency symbols.

Parameter: [TRMOUT](#)

Description: Suppresses all output messages to the terminal.

Syntax: [SET TRMOUT](#) = {[ON](#)|[OFF](#)}

where:

[ON](#)

Displays output messages to the terminal. This value is the default.

[OFF](#)

Suppresses messages to the terminal.

Parameter:	<code>UNITS</code>
Description:	<p>This parameter applies to StyleSheets.</p> <p>Specifies the unit of measure for page margins, column positions, and column widths.</p>
Syntax:	<p><code>SET UNITS = {<u>INCHES</u> CM PTS}</code></p> <p>where:</p> <p><u>INCHES</u></p> <p>Uses inches as the unit of measure. This value is the default.</p> <p>CM</p> <p>Uses centimeters as the unit of measure.</p> <p>PTS</p> <p>Uses points as the unit of measurement. (One inch = 72 points, one cm = 28.35 points)</p>
Parameter:	<code>USER</code>
Description:	<p>Enables user access to a data source or stored procedure protected by Information Builders security.</p>
Syntax:	<p><code>SET USER = <i>user</i></code></p> <p>where:</p> <p><i>user</i></p> <p>Is the user name that, with a password, enables access to a data source or stored procedure protected by Information Builders security.</p>

Parameter: [WEBTAB](#)

Description: Instructs FOCUS to enclose CRTFORM display fields in @ signs.

When the HTML/TP feature of Web390 generates replacement HTML forms for a 3270 screen, it can dynamically account for fields that may or may not be populated with data during execution. HTML/TP can use this technique with turnaround (T.) fields on CRTFORMs because they are enclosed in @ signs. These @-sign markers enable HTML/TP to recognize them and handle them dynamically on a customized HTML form. In contrast, CRTFORM display (D.) fields are not normally enclosed in @ signs.

Note: This setting is only for those MODIFY CRTFORM or Dialogue Manager -CRTFORM applications that will be used in conjunction with the HTML/TP feature of Web390. For information about Web390 and the HTML/TP feature, see the *Web390 for OS/390 and MVS Developer's Guide and Installation Manual*.

Syntax: `SET WEBTAB = {ON|OFF}`

where:

[ON](#)

Adds @ signs around CRTFORM display fields. These markers may cause the fields displayed on the CRTFORM to shift slightly to the right. Use this setting only for MODIFY CRTFORM or Dialogue Manager -CRTFORM applications that will be used in conjunction with the HTML/TP feature of Web390.

[OFF](#)

Does not place @ signs around CRTFORM display fields. This value is the default.

Parameter:	WEEKFIRST
Description:	This parameter is used in week computations by the HDIFF, HNAME, HPART, and HSETPT functions described in the <i>Using Functions</i> manual. The values from 1 to 7 represent Sunday through Saturday.
Syntax:	<pre>SET WEEKFIRST = number</pre> <p>where:</p> <p><i>number</i></p> <p>Is a number from one to seven, where one represents Sunday and seven represents Saturday. The U.S. English default value is seven (Saturday) meaning that Saturday is the first day of each week, so every Friday-Saturday transition is the start of a new week.</p> <p>The WEEKFIRST setting does not change the number that corresponds to each day of the week, it just specifies which one is considered the start of the week. The default of Saturday (7) as the first day of the week is consistent with the Microsoft SQL Server convention.</p>
Parameter:	WIDTH
Description:	Specifies the logical record length of your output data set when the STYLESHEET parameter is OFF. When the STYLESHEET parameter is in effect, FOCUS ignores the setting for WIDTH.
Syntax:	<pre>SET WIDTH = {n 130}</pre> <p>where:</p> <p><i>n</i></p> <p>Is the logical record length of your output data set. The default value is 130.</p>

Parameter:	XRETRIEVAL
Description:	Previews the format of a report without actually accessing any data. This parameter enables you to perform TABLE, TABLEF, or MATCH requests and produce HOLD Master Files without processing the report.
Syntax:	<code>SET XRETRIEVAL = {<u>ON</u> OFF}</code> where: <u>ON</u> Performs retrieval when previewing a report. This value is the default. <u>OFF</u> Specifies that no retrieval is to be performed.
Parameter:	YRTHRESH
Description:	Defines the start of a 100-year window globally or on a field-level. Used with DEFCENT, interprets the current century according to the given values. Two-digit years greater than or equal to YRTHRESH assume the value of the default century. Two-digit years less than YRTHRESH assume the value of one more than the default century. (See DEFCENT.) Note: This same result can be achieved by including the FDEFCENT and FYRTHRESH attributes in the Master File.
Syntax:	<code>SET YRTHRESH = {[<u>-</u>]yy <u>0</u>}</code> where: yy Is the year threshold for the window. If you do not supply a value, yy defaults to zero (0). If yy is a positive number, that number is the start of the 100-year window. Any two-digit years greater than or equal to the threshold assume the value of the default century. Two-digit years less than the threshold assume the value of one more than the default century. If yy is a negative number (-yy), the start date of the window is derived by subtracting that number from the current year, and the default century is automatically calculated. The start date is automatically incremented by one at the beginning of each successive year.

CHAPTER 2

Querying Your Environment

Topics:

- Using Query Commands
- List of Query Commands

You can query your environment to display information such as status of files, release information, server information, and joins.

Using Query Commands

Query commands display information about your metadata, physical data sources, language environment, and development and run-time environment.

Syntax

How to Issue a Query Command

? query [filename]

where:

query

Is the subject of the query.

filename

Is the name of the file that is the subject of the query. This parameter applies to only some queries.

To list the query commands, type a question mark in a procedure or at the command prompt.

Reference

Query Command Summary

The following is a list of query commands. A detailed description of each is in this topic.

? COMBINE	Lists FOCUS files comprising the current combined structures.
? DEFINE	Displays currently active virtual fields created by the DEFINE command or attribute.
? EUROFILE	Displays the currency data source in effect.
? F	Lists fields currently available to you.
? FDT	Displays physical attributes of a FOCUS data source.
? FF	Lists field names, aliases, and format information for an active Master File.
? FILE	Displays the number of segment instances in a FOCUS data source and the last time the data source was changed.
? FUNCTION	Displays the defined functions and their parameters.
? HOLD	Displays fields described in a HOLD Master File.
? HBUDGET	Displays the Hiperspace limits specified, and actual utilization statistics.
? JOIN	Displays join structures that exist between data sources.
? LANG	Displays information about National Language Support.
? LET	Displays word substitutions created with the LET command.
? LOAD	Provides information about all loaded files: the file type, file name and resident size.

? <i>n</i>	Displays an explanation of an error message (<i>n</i> represents the number of the error message).
? PFKEY	Displays the PF key assignments.
? PTF	Displays the PTFs applied to your version of FOCUS
? RELEASE	Displays the release number of your product.
? SET	Displays parameter settings that control your development and run-time environment.
? SET GRAPH	Displays parameter settings that control graphs produced with the GRAPH command.
? STAT	Displays statistics about the last command executed.
? STYLE	Displays the current settings for StyleSheet parameters.
? SU	Is communication available to the SU machine.
? USE	Displays data sources specified with the USE command.
? &&	Displays values of global variables.

Displaying Combined Structures

The ? COMBINE command displays files that are in the current combined structures.

Syntax

How to Display Combined Structures

```
? COMBINE [filename]
```

where:

filename

Is the data source containing the virtual fields. If *filename* is omitted, the command displays all virtual fields.

Example

Displaying Combined Structures

Issuing the command

```
? COMBINE
```

produces information similar to the following:

```
COMBINE EDUCFILE AND JOBFILE AS EDJOB
>
? COMBINE
FILE=EDJOB          TAG          PREFIX

    EDUCFILE
    JOBFILE
>
```

Displaying Virtual Fields

The ? DEFINE command lists the active virtual fields used in a request. The fields can be created by either the DEFINE command or DEFINE attribute in the Master File. The command displays field names of up to 32 characters. If a name exceeds 32 characters, then an ampersand (&) in the 32nd position indicates a longer field name.

Syntax

How to Display Virtual Fields

```
? DEFINE [filename]
```

where:

filename

Is the data source containing the virtual fields. If *filename* is omitted, the command displays all virtual fields.

Example

Displaying Virtual Fields

Assume that you created a virtual field named FULLNAME in a request against the EMPLOYEE database.

Issuing

```
? DEFINE
```

produces the following information:

FILE	FIELD NAME	FORMAT	SEGMENT	VIEW	TYPE
EMPLOYEE	PROJECTEDSAL	D12.2			
EMPLOYEE	FULLNAME	A26			

>

Reference

? DEFINE Query Information

The following information is listed for each virtual field created with DEFINE:

FILE	Is the name of the data source containing the virtual field.
FIELD NAME	Is the name of the virtual field.
FORMAT	Is the format of the virtual field. The notation is the same as that used for the FORMAT attribute in a Master File.
SEGMENT	Is the number of the segment in the Master File containing the virtual field. During reporting, your application treats the virtual field as a field in this segment. To relate segment numbers to segment names, use ? FDT.
VIEW	Is the root segment of DEFINE that specifies an alternate view. For example: <code>DEFINE FILE EMPLOYEE.JOBCODE</code>
TYPE	Indicates whether the virtual field is created by the DEFINE attribute in the Master File, or by a DEFINE command, identified by MASTER or a blank, respectively.

Displaying the Currency Data Source in Effect

The ? EUROFILE command displays the currency data source in effect.

Syntax How to Display the Currency Data Source in Effect

? EUROFILE

Example Displaying the Currency Data Source in Effect

Issuing the command

? EUROFILE

produces information similar to the following:

EUROFILE GBP

Displaying Available Fields

The ?F command displays the fields that are currently available.

?F displays entire 66 character field names.

Syntax How to Display Available Fields

?F filename

where:

filename

Is the name of a data source.

Example Displaying Available Fields

Issuing the command

?F EMPLOYEE

produces the following information:

```
FILENAME = EMPLOYEE
EMP_INFO  EMP_ID      LAST_NAME    FIRST_NAME   HIRE_DATE
DEPARTMENT CURR_SAL    CURR_JOBCODE ED_HRS
BANK_NAME  BANK_CODE  BANK_ACCT    EFFECT_DATE
DAT_INC    PCT_INC    SALARY       PAYINFO      JOBCODE
TYPE       ADDRESS_LN1 ADDRESS_LN2   ADDRESS_LN3  ACCTNUMBER
PAY_DATE   GROSS
DED_CODE   DED_AMT
JOBSEG     JOBCODE     JOB_DESC
SEC_CLEAR
SKILLS     SKILLS_DESC
DATE_ATTEND ATTENDSEG.EMP_ID
COURSE_CODE COURSE_NAME
```

Displaying the File Directory Table

The ? FDT command displays the file directory table, which lists the physical characteristics of a FOCUS data source.

A FOCUS data source is composed of fixed-length, 4096-byte records called pages. Each segment and each index (those fields designated by the keyword FIELDTYPE=I in the Master File) occupies an integral number of pages. The file directory table shows the amount of space occupied by each segment instance in a page, the starting and ending page numbers, and the number of pages in between for each segment and index.

Syntax

How to Display a File Directory Table

? FDT *filename*

where:

filename

Is the name of the data source.

Example

Displaying a File Directory Table

Issuing the command

? FDT EMPLOYEE

produces the following information:

DIRECTORY:EMPLOYEEFOCUS F ON 09/25/1997 AT 09.50.28								
DATE/TIME OF LAST CHANGE: 03/30/1999 16.19.22								
	SEGNAME	LENGTH	PARENT	START	END	PAGES	LINKS	TYPE
1	EMPINFO	22		1	1	1	6	
2	FUNDTRAN	10	1	2	2	1	2	
3	PAYINFO	8	1	3	3	1	3	
4	JOBSEG	11	3				4	
5	SECSEG	4	4				2	
6	SKILLSEG	11	4				2	
7	ADDRESS	19	1	4	4	1	2	
8	SALINFO	6	1	5	5	1	3	
9	DEDUCT	5	8	6	8	3	2	
10	ATTNDSEG	7	1				3	
11	COURSEG	11	10				2	
>								

Reference

? FDT Query Information

The following information is listed in the file directory table:

SEGNAME	Is the name of each segment in the file. The segments are also numbered consecutively down the left of the table. Unnumbered entries at the foot of the table are indexes, which belong to fields having the attribute FIELDTYPE=I in the Master File.
LENGTH	Is the length in words (units of four bytes) of each segment instance. Divide this number into 992 to determine the number of instances that can fit into a page.
PARENT	Is the parent segment. Each number refers to a segment name in the SEGNAME column.
START	Is the page number on which the segment or index begins.
END	Is the page number on which the segment or index ends.
PAGES	Is the number of pages occupied by the segment or index.
LINKS	Is the length, in words, of the pointer portion in each segment instance. Every segment instance consists of two parts, data and pointers. Pointers are internal numbers that are used to find other instances.
TYPE	Is the type of index. NEW indicates a binary index. OLD indicates a hash index. Segments of type KU, LM, DKU, DKM, KL, and KLU are not physically in this file; therefore, this information is omitted from the table.

Displaying Field Information for a Master File

The ?FF command displays field names, aliases, and format information for an active Master File.

Syntax

How to Display Field Information for a Master File

?FF *filename* [*string*]

where:

filename

Is the name of the Master File.

string

Is a character string up to 66 characters long. The command displays information only for fields beginning with the specified character string. If you omit this parameter, the command displays information for all fields in the Master File.

Example

Displaying Field Information for a Master File

Issuing the command

?FF EMPLOYEE

produces the following information:

```
FILENAME= EMPLOYEE
EMP_INFO
EMP_ID          EID          A9
LAST_NAME       LN           A15
FIRST_NAME      FN           A10
HIRE_DATE       HDT          16YMD
DEPARTMENT      DPT          A10
CURR_SAL        CSAL         D12.2M
CURR_JOBCODE    CJC          A3
ED_HRS          OJT          F6.2
BANK_NAME       BN           A20
BANK_CODE       BC           I6S
BANK_ACCT       BA           I9S
EFFECT_DATE     EDATE        16YMD
DAT_INC         DI           I6YMD
PCT_INC         PI           F6.2
SALARY          SAL          D12.2M
PAY_INFOJOBCODEJBC  A3
```

Displaying Data Source Statistics

The ? FILE command displays information such as the number of segment instances in a FOCUS data source and when the data source was last changed.

Syntax How to Display Data Source Statistics

? FILE *filename*
 where:
filename
 Is the name of the data source.

Example Displaying Data Source Statistics

Issuing the command

? FILE EMPLOYEE

produces statistics similar to the following:

STATUS OF FOCUS FILE: EMPLOYEEFOCUS A1 ON 03/12/99 AT 12.29.51					
SEGNAME	ACTIVE COUNT	DELETED COUNT	DATE OF LAST CHG	TIME OF LAST CHG	LAST TRANS NUMBER
EMPINFO	12		12/21/93	11.01.32	1
FUNDTRAN	6		11/16/89	16.19.19	12
PAYINFO	19		11/16/89	16.19.20	19
ADDRESS	21		11/16/89	16.19.21	21
SALINFO	70		11/16/89	16.19.22	448
DEDUCT	448		11/16/89	16.19.22	448
TOTAL SEGS	576				
TOTAL CHARS	8984				
TOTAL PAGES	8				
LAST CHANGE			01/29/96	11.01.32	1

Reference ? FILE Query Information

The following data source statistics are listed:

SEGNAME	Is the name of each segment in the data source. After the segments, the indexes are listed, if applicable. Indexes are those fields specified by the attribute FIELDTYPE=I in the Master File.
ACTIVE COUNT	Is the number of instances of each segment.
DELETED COUNT	Is the number of segment instances deleted, for which the space is not reused.
DATE OF LAST CHG	Is the date on which data in a segment instance or index was last changed.
TIME OF LAST CHG	Is the time of day, on a 24-hour clock, when the file's last update was made for that segment or index.
LAST TRANS NUMBER	Is the number of transactions performed by the last update request to access the segment. If the data source was changed under Simultaneous Usage mode, this column refers to the REF NUMB column of the CR HLIPRINT file.
TOTAL SEGS	Is the total number of segment instances in the file (shown under ACTIVE COUNT), and the number of segments deleted when the file was last changed (shown under DELETED COUNT).
TOTAL CHARS	Is the number of characters of data in the file.
TOTAL PAGES	Is the number of pages in the data source. Pages are physical records in FOCUS data sources. Each page is 4096 bytes.
LAST CHANGE	Is the date and time the data source was last changed.

If a data source is disorganized by more than 29%, that is, the physical placement of data in the data source is considerably different from its logical or apparent placement, the following message appears

```
FILE APPEARS TO NEED THE -REBUILD- UTILITY
REORG PERCENT IS A MEASURE OF FILE DISORGANIZATION
0 PCT IS PERFECT -- 100 PCT IS BAD
REORG PERCENT IS x%
```

where:

x

Is a percentage between 30 and 100.

The variable &FOCDISORG also indicates the level of disorganization. Following is an example of how you can use &FOCDISORG in a Dialogue Manager -TYPE command:

```
-TYPE THE AMOUNT OF DISORGANIZATION OF THIS FILE IS: &FOCDISORG
```

This command, depending on the amount of disorganization, produces a message similar to the following:

```
THE AMOUNT OF DISORGANIZATION OF THIS FILE IS: 10
```

When using a -TYPE command with &FOCDISORG, a message is displayed even if the percentage of disorganization is less than 30%.

Displaying Defined Functions

The ? FUNCTION command displays all defined functions and their parameters.

Syntax

How to Display DEFINE Functions

To display defined functions, issue the command:

```
? FUNCTION
```

Example

Displaying DEFINE Functions

Issuing the command

```
? FUNCTION
```

produces information similar to the following:

<u>Name</u>	<u>Format</u>	<u>Parameter</u>	<u>Format</u>
DIFF	D8	VAL1	D8
		VAL2	D8

Displaying HiperBudget Limits and Usage

The ? HBUDGET command displays the HiperSpace limits specified and actual utilization statistics, including: limits set at the system, server, user and file levels; the number of busy pages; the number of hiperextents allowed; and the ddnames and sizes of files allocated in hiperspace or spilled to disk.

Syntax

How to Display HiperBudget Limits and Usage

To display HiperBudget Limits and Usage, issue the command:

```
? HBUDGET
```

Example **Displaying HiperBudget Limits and Usage**

Issuing the command

```
? HBUDGET
```

produces information similar to the following:

```
Total system      limit is not set
Total server      limit is not set
Total hiperspace  limit is not set
Single file size  limit is  524288 pages
Total amount of busy pages is  616 pages
Number of extents is set to    127

DDname :Reserved :Hiperspace : Spilled :Spill DDn
```

Displaying HOLD Fields

The ? HOLD command lists fields described in a Master File created by the ON TABLE HOLD command. The list displays the field names, their aliases, and their formats as defined by the FORMAT (USAGE) attribute. The ? HOLD command displays field names up to 32 characters. If a field name exceeds 32 characters, an ampersand (&) in the 32nd position indicates a longer field name.

The ? HOLD command displays fields of a HOLD Master File created by the current request.

Syntax **How to Display HOLD Fields**

```
? HOLD [filename]
```

where:

filename

Is the name assigned in the AS phrase in the ON TABLE HOLD command. If you omit the file name, it defaults to HOLD.

Example **Displaying HOLD Fields**

Issuing the command

```
? HOLD
```

produces information similar to the following:

```
DEFINITION OF CURRENT HOLD FILE
FIELDNAME                                ALIAS      FORMAT

COUNTRY                                  E01        A10
CAR                                       E02        A16
```

Displaying JOIN Structures

The ? JOIN command lists the JOIN structures currently in effect. The command displays field names up to 12 characters. If a field name exceeds 12 characters, an ampersand in the twelfth position indicates a longer field name.

Syntax

How to Display JOIN Structures

? JOIN

Example

Displaying JOIN Structures

Issuing the command

? JOIN

produces information similar to the following:

JOINS CURRENTLY ACTIVE								
HOST			CROSSREFERENCE					
FIELD	FILE	TAG	FIELD	FILE	TAG	AS	ALL	WH
----	----	---	----	----	---	--	---	--
JOBCODE	EMPLOYEE		JOBCODE	JOBFILE			N	N

Reference

? JOIN Query Information

The following JOIN information is listed:

HOST FIELD	Is the name of the host field that is joining the data sources.
FILE	Is the name of the host data source.
TAG	Is a tag name used as a unique qualifier for field names in the host data source.
CROSSREFERENCE FIELD	Is the name of the cross-referenced field used to join the data sources.
FILE	Is the name of the cross-referenced data source.
TAG	Is a tag name used as a unique qualifier for field names in the cross-referenced data source.
AS	Is the name of the joined structure.
ALL	Displays Y for a non-unique join and N for a unique join.
WH	Displays Y for a conditional join, and N for an equi-join.

Displaying National Language Support

The ? LANG command displays information about National Language Support.

Syntax **How to Display Information About National Language Support**
To display information about National Language Support:
`? LANG`

Example **Displaying Information About National Language Support**
Issuing the command
`? LANG`
produces information similar to the following:

```
LANGUAGE AND DBCS STATUS

Language           01/AMENGLISH   (    )
Code Page          00037
Dollar value       5B($ )
DBCS Flag          OFF(SBCS)
```

Displaying LET Substitutions

The ? LET command lists the active word substitutions created by the LET command. A word in the left column is used in a report request to represent the word or phrase in the right column. For more information on the LET command, see your documentation on defining LET substitutions.

Syntax **How to Display LET Substitutions**
`? LET`

Example **Displaying LET Substitutions**
Issuing the command
`? LET`
produces information similar to the following:

```
PR                PRINT
TF                TABLE FILE EMPLOYEE
```


Displaying Information About Loaded Files

The ? LOAD command displays the file type, file name, and resident size of currently loaded files.

Syntax

How to Display Information About Loaded Files

```
? LOAD [filetype]
```

where:

filetype

Specifies the type of file (MASTER, FOCEXEC, Access File, FOCCOMP, or MODIFY) on which information will be displayed. To display information on all memory-resident files, omit file type.

Example

Displaying Information About Loaded Files

Issuing the command

```
? LOAD
```

produces information similar to the following:

```
FILES CURRENTLY LOADED
```

CAR	MASTER	4200	BYTES
EXPERSON	MASTER	4200	BYTES
CARTEST	FOCEXEC	8400	BYTES

Displaying Explanations of Error Messages

The ? n command displays a detailed explanation of an error message, providing assistance in correcting the error.

Error messages generated by certain data adapters, such as the DB2 and MODEL 204 data adapters, are also accessible through this feature.

Syntax

How to Display Explanations of Error Messages

```
? n
```

where:

n

Is the error message number.

Example **Displaying Explanations of Error Messages**

If you receive the message

```
(FOC125) RECAP CALCULATIONS MISSING
```

issuing the command

```
? 125
```

produces the following message:

```
(FOC125) RECAP CALCULATIONS MISSING
The word RECAP is not followed by a calculation. Either the RECAP should
be removed, or a calculation provided.
```

Displaying PF Key Assignments

The ? PFKEY command displays the PF key assignments.

Syntax **How to Display PF Key Assignments**

To display the PF key assignments, issue the command:

```
? PFKEY
```

Example **Displaying PF Key Assignments**

Issuing the command

```
? PFKEY
```

produces results similar to the following:

PF01 = HX	PF02 = CANCEL	PF03 = END	PF04 = RETURN
PF05 = RETURN	PF06 = SORT	PF07 = BACKWARD	PF08 = FORWARD
PF09 = RETURN	PF10 = LEFT	PF11 = RIGHT	PF12 = UNDO
PF13 = RETURN	PF14 = RETURN	PF15 = END	PF16 = RETURN
PF17 = RETURN	PF18 = RETURN	PF19 = BACKWARD	PF20 = FORWARD
PF21 = RETURN	PF22 = RETURN	PF23 = RETURN	PF24 = UNDO

Querying Which PTFs Have Been Applied for a Specific Release

The ? PTF command displays a list of PTFs that have been applied to the version of FOCUS you are currently using.

Syntax **How to Query a List of PTFs**

```
? PTF
```

Example

Querying a List of PTFs

Issuing the command

```
? PTF
```

produces results similar to the following:

```
>
? ptf
PTFS APPLIED TO RELEASE /0XFOU
FROM PTFABLE LOCATED IN IBTEST LOADLIB C1
```

COUNT	PTF NUM	CREATED	APPLIED	SUPERSEDED BY	PTF LEVEL
1)	95020	112000
2)	107104	112000
3)	110703	112000
4)	112000	19990427	19990513	200295

Note: Dots are used to denote the lack of data if no information exists for a column entry in the resulting report. If there are no PTFs for the version of FOCUS that you are currently running, the following is displayed:

```
NO PTFS HAVE BEEN APPLIED
```

Displaying the Release Number

The ? RELEASE command displays the number of the currently installed release of your product.

Syntax

How to Display the Release Number

```
? RELEASE
```

Example

Displaying the Release Number

Issuing the command

```
? RELEASE
```

produces information similar to the following:

```
FOCUS 7.2.0 created 11/07/2001 11.38.32
```

Displaying Parameter Settings

The ? SET command lists the parameter settings that control your development and run-time environments. Your application sets default values for these parameters, but you can change them with the SET command.

SET parameters are described in Chapter 1, *Customizing Your Environment*.

Syntax

How to Display Parameter Settings

```
? SET [ALL|[FOR] parameter]
```

where:

ALL

Displays all possible parameter settings.

parameter

Is a SET parameter. This displays the setting for the specific parameter.

FOR

Includes where the parameter can be set from in addition to the parameter setting.

Example

Displaying Parameter Settings

Issuing the command

```
? SET
```

produces information similar to the following:

PARAMETER SETTINGS					
ALL.	OFF	HIPERFOCUS	OFF	QUALCHAR	.
ASNAMES	FOCUS	HOLDATTRS	FOCUS	QUALTITLES	OFF
AUTOINDEX	ON	HOLDLIST	ALL	RECAP-COUNT	OFF
AUTOPATH	ON	HOLDSTAT	OFF	SAVEMATRIX	ON
BINS	64	HOTMENU	OFF	SCREEN	ON
BLKCALC	NEW	INDEX TYPE	NEW	SHADOW PAGE	OFF
BYPANELING	OFF	LANGUAGE	AMENGLISH	SPACES	AUTO
CACHE	0	LINES/PAGE	66	SQLENGINE	
CARTESIAN	OFF	LINES/PRINT	57	TCPIPINT	OFF
CDN	OFF	MESSAGE	ON	TEMP DISK	A
COLUMNSCROLL	OFF	MODE	CMS	TERMINAL	IBM3270
DATETIME	STARTUP/RESET	NODATA	.	TITLES	ON
DEFCENT	19	PAGE-NUM	ON	WIDTH	130
EMPTYREPORT	OFF	PANEL	0	WINPFKEY	OLD
EXTSORT	ON	PAUSE	ON	XRETREIVAL	ON
FIELDNAME	NEW	PRINT	ONLINE	YRTHRESH	0
FOCSTACK SIZE	8	PRINTPLUS	ON		

Some parameters are listed differently from the way you specify them in the SET command. These include:

FOCSTACK SIZE	Is the same as the FOCSTACK parameter.
INDEX TYPE	Is the same as the INDEX parameter.
LINES/PAGE	Is the same as the PAPER parameter.
LINES/PRINT	Is the same as the LINES parameter.
SHADOW PAGES	Is the same as the SHADOW parameter.

Example

Displaying a Single Parameter Setting

Issuing the command

```
? SET ONLINE-FMT
```

produces the following if the parameter is set to its default value:

```
ONLINE-FMT          STANDARD
```

Example

Displaying Where a Parameter Can Be Set

Issuing the command

```
? SET FOR EXTSORT
```

produces the following information:

```
EXTSORT              ON
```

```
-----
SETTABLE FROM COMMAND LINE      : YES
SETTABLE ON TABLE              : YES
SETTABLE FROM SYSTEM-WIDE PROFILE : YES
SETTABLE FROM HLI PROFILE       : YES
POOL TABLE BOUNDARY           : YES
>
```

Displaying Parameters That Cannot Be Set in an Area

The ? SET NOT command produces a list of SET commands that cannot be set in a specific area. The areas for which you can find this information are the PROMPT command, report requests, the FOCPARM profile, the HLI profile, and Pooled Tables.

Syntax

How to Determine Where a Command Is Valid

? SET NOT *area*

where:

area

- Is one of the following:
- PROMPT is the PROMPT command.
- ONTABLE is a report request.
- FOCPARM is the FOCPARM profile.
- HLIPROF is the HLI profile.
- PT is in Pooled Tables.

Example

Displaying Parameters That Cannot Be Set With ONTABLE

Issuing the command

? SET NOT ONTABLE

produces the following information:

NON-SETTABLE ON TABLE PARAMETER SETTINGS					
BINS	64	LANGUAGE	AMENGLISH	REBUILDMSG	1000
BLKCALC	NEW	MAXPOOLMEM	32768	SAVEMATRIX	ON
BYPANELING	OFF	MDIBINS	8000	TCPIPINT	OFF
CACHE	0	MDIPROGRESS	100000	TEMP DISK	C
COLUMNSCROLL	OFF	MODE	CMS	TRMSD	24
DATEDISPLAY	OFF	MPRINT	NEW	TRMSW	80
DATEFNS	ON	POOL	OFF	TRMTYP	1 (3270)
DEFCENT	19	POOLBATCH	OFF	WEBHOME	OFF
EUROFILE		POOLFEATURE	OFF	WIDTH	130
FIELDNAME	NEW	POOLMEMORY	16384	WINPFKEY	OLD
FOCSTACK SIZE	8	POOLRESERVE	1024	YRTHRESH	0
HTMLMODE	OFF	PRINTPLUS	OFF		

Displaying Graph Parameters

The ? SET GRAPH command lists the parameter settings that control graphs produced with the GRAPH command. These parameters are described further in Chapter 1, *Customizing Your Environment*.

Syntax

How to Display Graph Parameters

? SET GRAPH

Example

Displaying Graph Parameters

Issuing the command

? SET GRAPH

produces information similar to the following:

GRAPH PARAMETER SETTINGS

AUTOTICK	ON	HISTOGRAM	ON
BARNUMB	OFF	HMAX	.00
BARSPEACE	0	HMIN	.00
BARWIDTH	1	HSTACK	OFF
BSTACK	OFF	HTICK	.00
DEVICE	IBM3270	PIE	OFF
GMISSING	OFF	VAUTO	ON
GMISSVAL	.00	VAXIS	66
GPROMPT	OFF	VCLASS	.00
GRIBBON(GCOLOR)	OFF	VGRID	OFF
GRID	OFF	VMAX	.00
GTREND	OFF	VMIN	.00
HAUTO	ON	VTICK	.00
HAXIS	130	VZERO	OFF
HCLASS	.00		

>

If you change the PLOT parameter settings, a small table appears at the end of the list:

PLOT TABLE (EBCDIC):

ENTER PLOT MODE	0050 (FOR 3284 WIDTH)
EXIT PLOT MODE	0018 (FOR 3284 HEIGHT)
LEFT	0000
RIGHT	0000
UP	0000
DOWN	0000

The entries in the table at the bottom are:

ENTER PLOT MODE Width of graph on IBM 3284 or 3287 printer.

EXIT PLOT MODE Height of graph on IBM 3284 or 3287 printer.

Ignore the parameters LEFT, RIGHT, UP, and DOWN.

Displaying Command Statistics

The ? STAT command lists statistics for the most recently executed command. Each statistic applies only to a certain command. If another command is executed, the statistic is either 0 or does not appear in the list at all. When you execute commands in procedures, these statistics are automatically stored in Dialogue Manager statistical variables. For more information, see Chapter 3, *Managing an Application With Dialogue Manager*.

Syntax

How to Display Command Statistics

? STAT

Example

Displaying Command Statistics

Issuing the command

? STAT

produces information similar to the following:

STATISTICS OF LAST COMMAND					
RECORDS	=	0	SEGS DELTD	=	0
LINES	=	0	NOMATCH	=	0
BASEIO	=	0	DUPLICATES	=	0
SORTIO	=	0	FORMAT ERRORS	=	0
SORT PAGES	=	0	INVALID CONDTs	=	0
READS	=	0	OTHER REJECTS	=	0
TRANSACTIONS	=	0	CACHE READS	=	0
ACCEPTED	=	0	MERGES	=	0
SEGS INPUT	=	0	SORT STRINGS	=	0
SEGS CHNGD	=	0	INDEXIO	=	0
INTERNAL MATRIX CREATED: YES			AUTOINDEX USED: NO		
SORT USED: FOCUS			AUTOPATH USED: NO		
AGGREGATION BY EXT.SORT: NO			HOLD FROM EXTERNAL SORT: NO		
>					

Reference

? STAT Query Information

The following information displays:

RECORDS	Is for TABLE, TABLEF, and MATCH commands. It indicates the number of data source records used in the report. The meaning of a record depends on the type of data source used.
LINES	Is for TABLE and TABLEF commands. It indicates the number of lines displayed in a report.
BASEIO	Is for TABLE, TABLEF, GRAPH, MODIFY, and FSCAN commands. It indicates the number of I/O operations performed on the data source.

<code>SORTIO</code>	Is for TABLE, TABLEF, GRAPH, and MATCH commands. It indicates the number of I/O operations performed on the FOCSORT file, which is a work file invisible to the end user.
<code>SORTPAGES</code>	Is for TABLE and TABLEF commands. It indicates the number of physical records in the FOCSORT file.
<code>READS</code>	Is for the MODIFY and FSCAN commands. It indicates the number of fixed format records read in external files by the FIXFORM command.
<code>TRANSACTIONS</code>	Is for the MODIFY and FSCAN commands. It indicates the number of transactions processed.
<code>ACCEPTED</code>	Is for the MODIFY and FSCAN commands. It indicates the number of transactions accepted.
<code>SEGS INPUT</code>	Is for the MODIFY and FSCAN commands. It indicates the number of segment instances accepted in the data source.
<code>SEGS CHNGD</code>	Is for the MODIFY and FSCAN commands. It indicates the number of segment instances updated in the data source.
<code>SEGS DELTD</code>	Is for the MODIFY and FSCAN commands. It indicates the number of segment instances deleted from the data source.
<code>NOMATCH</code>	Is for the MODIFY and FSCAN commands. It indicates the number of transactions rejected for lack of matching values in the data source. This occurs on an ON NOMATCH REJECT condition.
<code>DUPLICATES</code>	Is for the MODIFY and FSCAN commands. It indicates the number of transactions rejected because their matching field values already exist in the data source. This occurs on an ON MATCH REJECT condition.
<code>FORMAT ERRORS</code>	Is for the MODIFY and FSCAN commands. It indicates the number of transactions rejected because field values for data fields do not conform to the field formats defined in the Master File.
<code>INVALID CONDTs</code>	Is for the MODIFY and FSCAN commands. It indicates the number of transactions rejected because their values failed validation tests.
<code>OTHER REJECTS</code>	Is for the MODIFY and FSCAN commands. It indicates the number of transactions rejected for reasons other than those listed above.
<code>CACHE READS</code>	Is the number of cache reads performed. For more information on the cache, see Chapter 1, <i>Customizing Your Environment</i> .
<code>MERGES</code>	Is the number of times that merge routines were invoked.
<code>SORT STRINGS</code>	Is the number of times that the internal sort capacity was exceeded.

INTERNAL MATRIX CREATED	Indicates how report sorting was handled. If an external sort handled it entirely, the value is NO; if both the application and an external sort handled it, the value is Y.
SORT USED	Is the type of sort facility used. It can have a value of FOCUS, EXTERNAL, SQL, or NONE. NONE means that the report did not require sorting.
AGGREGATION BY EXT. SORT	Uses external sorts to perform aggregation.
AUTOINDEX USED	Automatically takes advantage of indexed fields to speed data retrieval.
AUTOPATH USED	Selects an optimal retrieval path for accessing a data source.
HOLD FROM EXTERNAL SORT	Creates hold files with an external sort.

Displaying StyleSheet Parameter Settings

The ? STYLE command displays the current settings for StyleSheet parameters.

Syntax

How to Display StyleSheet Parameter Settings

? [SET] STYLE

Example

Displaying StyleSheet Parameter Settings

Issuing the command

? STYLE

produces information similar to the following:

ONLINE-FMT	
OFFLINE-FMT	STANDARD
STYLESHEET	ON
SQUEEZE	OFF
PAGESIZE	LETTER
ORIENTATION	PORTRAIT
UNITS	INCHES
LABELPROMPT	OFF
LEFTMARGIN	.250
RIGHTMARGIN	.250
TOPMARGIN	.250
BOTTOMMARGIN	.250
STYLEMODE	FULL
TARGETFRAME	
FOCEXURL	
BASEURL	

Reference

? STYLE Query Information

The following StyleSheet information is listed:

ONLINE-FMT	Is the format of report output. Can produce output as HTML or as un-styled character-based output.
OFFLINE-FMT	Is the format of report output. Can produce output as HTML or as un-styled character-based output.
STYLESHEET	Rejects or accepts StyleSheet parameters that specify formatting options such as page size, orientation, and margins.
LABELPROMPT	Specifies on which label of the first page to begin printing a multi-pane report, such as a mailing label report.
STYLEMODE	Speeds the retrieval of large report output by displaying output in multiple HTML tables where each table is a separate report page.
ORIENTATION	Is the page orientation for styled reports. Can be either portrait or landscape.
UNITS	IS the unit of measure for PostScript and PDF report output, as either inches, centimeters, or points.
TOPMARGIN	Is the top boundary for a page of report output.
BOTTOMMARGIN	Is the bottom boundary for a page of report output.
LEFTMARGIN	Is the left boundary for a page of report output.
RIGHTMARGIN	Is the right boundary for a page of report output.
TARGETFRAME	Is a frame to which all drill-down hyperlinks are directed.
FOCEXURL	Is the path for a procedure stored on the Web server and run through the WebFOCUS CGI.
BASEURL	Is the default location where the browser searches for relative URLs specified in the HTML documents created by your application.

Displaying Information About the SU Machine

The ? SU command displays the communication available to the FOCUS Database Server.

Syntax **How to Display Information About the FOCUS Database Server**

```
? SU [userid|ddname]
```

where:

```
userid
```

Is a valid user ID.

```
ddname
```

Is a valid ddname.

Example **Displaying Information About the FOCUS Database Server**

Issuing the command

```
? SU SYNCA
```

produces the following information:

USERID	FILEID	QUEUE
WIBMLH	QUERY	
WIBJBP	CAR	

Displaying Data Sources Specified With USE

The ? USE command displays data sources specified with the USE command.

Syntax **How to Display Data Sources Specified With USE**

```
? USE
```

Example **Displaying Data Sources Specified With USE**

Issuing the command

```
? USE
```

produces information similar to the following:

DIRECTORIES IN USE ARE:		
CAR	FOCUS	F
EMPLOYEE	FOCUS	F
LEDGER	FOCUS	F

Displaying Global Variable Values

The ? && command lists Dialogue Manager global variables and their current values. Global variables maintain their values for the duration of the stored procedure.

See your documentation about Dialogue Manager for details on global and other variables.

Syntax

How to Display Global Variable Values

? &&

Your site may replace the ampersand (& or &&) indicating Dialogue Manager variables, with another symbol. In that case, use the replacement symbol in your query command. For example, if your installation uses the percent sign (%) to indicate Dialogue Manager variables, list global variables by issuing:

? %%

You can query all Dialogue Manager variables (local, global, and system) from a stored procedure by issuing:

-? &

Example

Displaying Global Variable Values

Issuing the command

? &&

produces information similar to the following:

```
&&STORECODE    001
&&STORENAME    MACYS
```

CHAPTER 3

Managing an Application With Dialogue Manager

Topics:

- Overview of Dialogue Manager Capabilities
- Creating and Storing Procedures
- Executing Procedures
- Including Comments in a Procedure
- Overview of Dialogue Manager Commands
- Sending a Message to the User: -TYPE
- Controlling Execution: -RUN, -EXIT, and -QUIT
- Branching
- Looping
- Using Expressions: -SET
- Additional Facilities
- Using Variables in Procedures
- Supplying Values for Variables at Run Time
- Dialogue Manager Quick Reference

This topic describes how to make report procedures more dynamic by using Dialogue Manager control commands and variables.

Overview of Dialogue Manager Capabilities

Dialogue Manager enables you to execute stored procedures. In the FOCUS community, stored procedures are referred to as FOCEXECs. In this topic, they are referred to simply as *procedures*.

Dialogue Manager helps you build and manage the execution of procedures, giving you flexibility in application design. You can use Dialogue Manager control commands to determine the sequence in which FOCUS commands (such as TABLE) execute. Dialogue Manager also enables you to use variables in your procedures and supply values for those variables at run time. You can create a dialogue between the user and the terminal through various prompting methods, including full-screen forms, menus and windows that you design yourself, and system queries, as well as supplying values directly in the procedure.

Using Dialogue Manager control commands and variables, your application can respond to user input and environment conditions at run time. It is important to understand how Dialogue Manager processes an application's commands and variables.

Example

Processing a Procedure

The following example traces the execution process of a procedure. The numbers at the left refer to explanatory notes that follow the example.

```
1.  -TOP
2.  -PROMPT &WHICHCITY. ENTER NAME OF CITY OR DONE.
3.  -IF &WHICHCITY EQ 'DONE' GOTO QUIT;
4.  TABLE FILE SALES
      SUM UNIT_SOLD
      BY PROD_CODE
      IF CITY IS &WHICHCITY
      END
5.  -RUN
6.  -GOTO TOP
7.  -QUIT
```

Assume that this procedure is stored in a file named SLRPT. To execute it, the user types either of the following:

```
EXEC SLRPT
```

or

```
EX SLRPT
```

The following describes the individual steps of the procedure:

1. `-TOP`

This is a label, which serves as a target to which `-IF ... GOTO` or `-GOTO` commands transfer processing control. Labels themselves call for no special processing, so in this case control passes to the next command.

2. `-PROMPT &WHICHCITY. ENTER NAME OF CITY OR DONE.`

The prompt “ENTER NAME OF CITY OR DONE” appears on the terminal. Assume the user types “STAMFORD” and the variable value is stored for later use. Processing continues with the next line.

3. `-IF &WHICHCITY EQ 'DONE' GOTO QUIT;`

Had DONE been entered, control would pass to `-QUIT` at the bottom of the procedure. This would end processing, cause an immediate exit from this procedure, and return control to the FOCUS prompt. Since STAMFORD was entered, processing continues with the next line.

4. `TABLE FILE SALES`

`.
.
.`

Since there is no leading hyphen, this is interpreted as a FOCUS command. Only Dialogue Manager commands execute immediately, so the next five lines are placed in the stack where FOCUS commands are kept until executed; this is referred to as FOCSTACK. Note that the value STAMFORD, entered in response to the prompt, is inserted into the FOCUS command line as the value for `&WHICHCITY`.

At this point the FOCSTACK looks like this:

```
TABLE FILE SALES
SUM UNIT_SOLD
BY PROD_CODE
IF CITY IS STAMFORD
END
```

Control passes to the next Dialogue Manager command.

5. `-RUN`

This command sends the stack to FOCUS, which executes the stored request and returns control to the next Dialogue Manager command.

6. `-GOTO TOP`

Control is now routed back to `-TOP`, thus establishing a loop. Execution continues from `-TOP` with the `-PROMPT` command.

7. -QUIT

This command is reached when the user types DONE in response to the prompt. The procedure is exited and the FOCUS prompt appears.

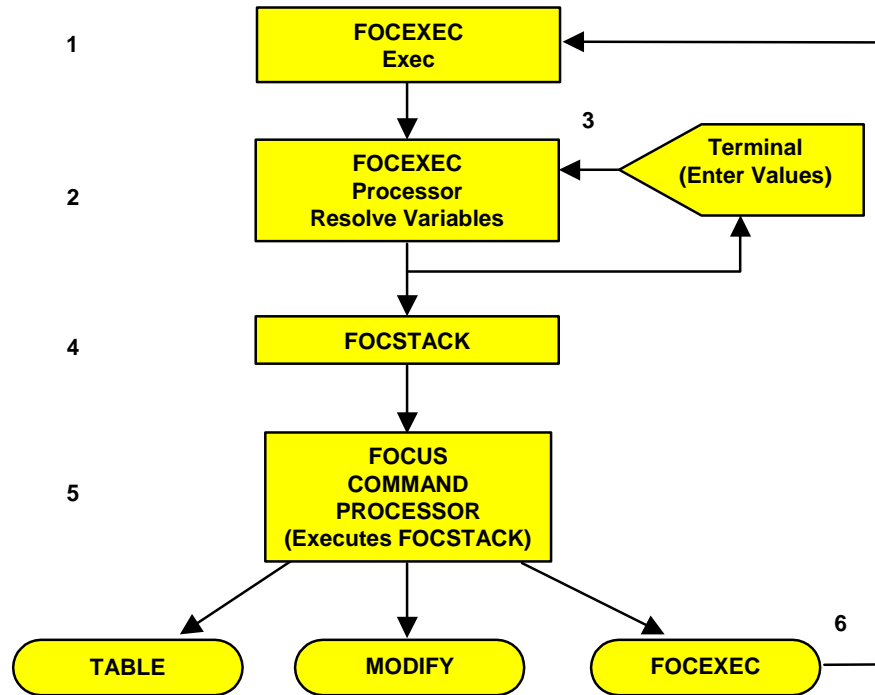


Figure 3-1. Schematic Diagram of Procedure Processing

1. Processing begins from the command processor when a procedure is invoked for execution at the FOCUS prompt (for example, EX SLRPT).
2. The FOCEXEC Processor reads each line of the procedure. Any variables on the line are assigned their current values.
3. If a variable is missing a value, FOCUS issues a prompt. The user then supplies the missing value.
4. When a command line that contains no Dialogue Manager commands is fully expanded with any variables resolved (through either a -SET command or prompting), it is placed onto the command execution stack (FOCSTACK).
5. Dialogue Manager execution commands (for example, -RUN) and statistical variables flush the FOCSTACK and route all currently stacked commands to the FOCUS Command Processor.

6. In the previous example the FOCUS Command Processor routes execution to the TABLE module and executes the TABLE request that was stacked.

By the time your FOCSTACK is ready for execution, this has happened:

- All variables have received values and these values have been integrated into the command lines containing variables.
- Dialogue Manager commands have been used to place FOCUS commands into proper sequential order for execution.
- At this point the FOCUS Command Processor no longer sees any Dialogue Manager commands. It only sees FOCUS command lines in the stack.

Note: Any FOCUS command can be placed in a procedure. This includes the EXEC command itself. When an EXEC command is processed in a procedure, the commands from the new procedure are first stacked and then executed. Multiple levels of nesting are permitted when you use the EXEC command, while only four levels of nesting are permitted when you use -INCLUDE.

Overview of Dialogue Manager Variables

You can write procedures that contain variables whose values are unknown until run time; this technique allows a user to customize the procedure by supplying different values each time it executes. Variables fall into two categories:

- Local and global variables, whose values must be supplied at run time. Local variables retain their values only for one procedure. Global variables retain their values across procedures unless you explicitly clear them. They lose their values when you exit from FOCUS. You create a local variable by choosing a name that starts with a single ampersand (&); you create a global variable by choosing a name that starts with a double ampersand (&&).
- System and statistical variables, whose values are automatically supplied by the system when a procedure references them. System and statistical variables have names that begin with a single ampersand (&). For example, the variable &LINES indicates how many lines of output were produced, and the variable &DATE indicates the current date.

For complete information about variables, see *Using Variables in Procedures* on page 3-49.

Creating and Storing Procedures

You can create procedures with your system editor or with the FOCUS integrated text editor, TED. TED has two features that make it particularly useful for creating and editing procedures:

- If you type TED without specifying a procedure name, the last executed procedure is automatically selected. This is convenient when developing and testing new procedures.
- You can test the execution of the procedure by typing RUN on the TED command line. This automatically saves the procedure and executes it. If there is an error in your procedure, type TED. This brings you back into the editor and places you directly on the line in which the error was detected.

These options complement the FILE and SAVE options that are common to other editors.

Follow these general rules when you are creating procedures:

- Dialogue Manager commands must begin in the first position of the line.
- At least one space must be inserted between the Dialogue Manager command and other text.
- If a Dialogue Manager command exceeds one line, the following line must begin with a hyphen (-) in the first position. The continuation line can begin immediately after the hyphen, or you may insert a space between the hyphen and the rest of the line.
- Procedures must have the record format RECFM=F and the logical record length (LRECL) 80.

Executing Procedures

Procedures are generally initiated from the FOCUS prompt (>). Type the command EXEC, or its abbreviation EX, followed by the name of the procedure.

Example

Executing a Procedure

Either of the following commands

```
EXEC SLRPT
```

or

```
EX SLRPT
```

summons the procedure named SLRPT for execution.

Controlling Access to Data

You can set a password in a procedure and tie it to different portions of a procedure.

Syntax

How to Set a Password in a Procedure

```
-PASS password
```

where:

password

Is a password or a variable containing a password.

Since -PASS is a Dialogue Manager command, it executes immediately and is not sent to the FOCSTACK. This means that the user need not issue the password with the SET command.

Including Comments in a Procedure

It is good practice to include comments in procedures for the benefit of others who may read or refine them at a later date. Comments are particularly recommended as procedure headers to give version, date, and other relevant information. It is easier to track and maintain large software applications when they are carefully commented. Comments are ignored during actual execution.

To add comment lines to a Dialogue Manager procedure, precede them with a hyphen and an asterisk (-*). Any text whatsoever may immediately follow the -*. You can place comment lines anywhere in a procedure.

Comments do not appear on the terminal nor do they trigger any processing. They are visible only when viewing the contents of the procedure through the editor and are strictly for the benefit of the developer. However, you can view comments on the terminal by using the option ECHO = ALL.

Example

Including Comments in a Procedure

The following example contains two comment lines:

```
-* Version 1 6/30/85 SLRPT FOCEXEC
-* Component of Retail Sales Reporting Module
TABLE FILE SALES
HEADING CENTER
"MONTHLY SALES FOR STAMFORD"
.
.
.
```

Overview of Dialogue Manager Commands

Dialogue Manager provides commands for accomplishing the following tasks:

- Sending messages to the user.
- Displaying values.
- Controlling the values of variables, including reading variables from and writing values to an external file.
- Testing conditions and branching.
- Controlling the execution of stacked commands.
- Calling another procedure.
- Issuing operating system commands specific to your environment.

Reference

Summary of Dialogue Manager Commands

The following pages describe all Dialogue Manager commands. They are listed in alphabetical order. The categories used to describe them in the quick reference at the end of this topic are briefly outlined below:

Command	Lists the name of the command.
Syntax	Shows exactly how the command components must appear in a procedure.
Function	Outlines the meaning and purpose of the command.
Similar Command	Describes the relationship between the Dialogue Manager command and other FOCUS commands (for example, -TYPE and TYPE).

Command	Meaning
- *	Is a comment line; it has no action.
-CLOSE ddname	Closes the specified -READ or -WRITE file.
-CLOSE *	Closes all -READ and -WRITE files currently open.
-CMS	Executes a CMS command from within Dialogue Manager.
-CMS RUN	In CMS, loads and executes a user-written function.
-CRTCLEAR	Clears the screen display.
-CRTFORM	Initiates full-screen variable data entry.

Command	Meaning
<code>-DEFAULT</code> <code>-DEFAULTS</code>	Presets initial values for variable substitution.
<code>-EXIT</code>	Executes stacked commands and returns to the FOCUS prompt.
<code>-GOTO</code>	Establishes an unconditional branch.
<code>-HTMLFORM</code>	For use with the Web Interface to FOCUS.
<code>-IF</code>	Tests and branches control based on test results.
<code>-INCLUDE</code>	Dynamically incorporates one procedure in another.
<code>-label</code>	Is a user-supplied name that identifies the target for <code>-GOTO</code> or <code>-IF</code> .
<code>-MVS RUN</code>	Same as <code>-TSO RUN</code> .
<code>-PASS</code>	Sets password directly.
<code>-PROMPT</code>	Types a prompt message on the screen and reads a reply.
<code>-QUIT</code>	Exits the procedure without executing it.
<code>-READ</code>	Reads records from a non-FOCUS file.
<code>-REPEAT</code>	Executes a loop.
<code>-RUN</code>	Executes all stacked FOCUS commands and returns to procedure for further processing.
<code>-SET</code>	Assigns a value to a variable.
<code>-TSO RUN</code>	In MVS/TSO, loads and executes a user-written function.
<code>-TYPE</code>	Types informative message to screen or other output device.
<code>-WINDOW</code>	Invokes Window Painter, transferring control from the procedure to the specified window file.
<code>-WRITE</code>	Writes a record to a non-FOCUS file.
<code>- "..."</code>	Brackets contents for <code>-CRTFORM</code> display line.
<code>-? SET SETCOMMAND &myvar</code>	Captures the setting of SETCOMMAND in &myvar.
<code>-? &[string]</code>	Displays the values of currently defined amper variables.

Sending a Message to the User: -TYPE

The Dialogue Manager command -TYPE enables you to send informative messages to the screen while a procedure is processing. These messages serve a variety of functions. They can explain the purpose of the procedure, the results of computations or calculations, or preface prompts requesting information from the terminal. -TYPE triggers these messages.

Syntax

How to Send a Message to the User

```
-TYPE[+] text  
-TYPE[0] text  
-TYPE[1] text
```

where:

+

Suppresses the line feed following the printing of text.

0

Forces a line feed before the message text is displayed.

1

Forces a page eject before the message text is printed.

text

Is all succeeding text including variable values supplied on the same command line. It sends the text to the screen, followed by a line feed. It remains on screen until scrolled off or replaced by a new screen.

The options +, 0, and 1 are used to pass printer control characters to the output device and are particularly useful for character printers. Options + and 1 do not work on IBM 3270-type terminals. -TYPE sends the text to the terminal as soon as it is encountered in the processing of a procedure.

Example

Sending a Message to the Users

The following is an example of the use of -TYPE:

```
-* Version 1 6/30/85 SALERPT FOCXEC  
-TYPE This report calculates percentage of returns  
TABLE FILE SALES  
.  
.  
.
```

Note: The -TYPE message need not be enclosed in quotation marks, since FOCUS understands that -TYPE signals a following textual message. If you use quotation marks, they will appear along with the message. This differs from the use of TYPE in MODIFY, where quotation marks are used as delimiters and must enclose informative text.

Controlling Execution: -RUN, -EXIT, and -QUIT

Dialogue Manager enables you to manage the flow of execution with these commands:

- -RUN executes stacked commands and continues the procedure.
- -EXIT executes stacked commands and exits the procedure.
- -QUIT cancels execution and exits the procedure.

Executing Stacked Commands and Continuing the Procedure: -RUN

The Dialogue Manager command -RUN causes immediate execution of all stacked FOCUS commands and closes any external files opened with -READ or -WRITE. Following execution, processing of the procedure continues with the line that follows -RUN.

Example

Executing Stacked Commands and Continuing the Procedure

The following example illustrates the use of -RUN to execute stacked code and then return to the procedure.

```
1. -TYPE This report calculates percentage of returns.
2. TABLE FILE SALES
   .
   .
   .
   END
3. -RUN
4. -TYPE This routine reports on data in the employee file.
   TABLE FILE EMPLOYEE
   .
   .
   .
   END
```

The procedure processes as follows:

1. The command -TYPE generates a message.
2. The FOCUS code is stacked.
3. The command -RUN causes the stacked commands to be executed and the output returned.
4. Processing continues with the line following -RUN. In this case, another message is sent and another TABLE request is initiated.

Executing Stacked Commands and Exiting the Procedure: -EXIT

-EXIT forces execution of stacked FOCUS commands as soon as it is encountered. However, instead of returning to the procedure, -EXIT closes all external files, terminates the procedure, and, either returns you to the FOCUS prompt or to the calling procedure.

Example

Executing Stacked Commands and Exiting the Procedure

In the following example, either the first TABLE request or the second TABLE request executes, but not both:

```
1. -TYPE This report calculates percentage of returns.
2. -IF &PROC EQ 'EMPLOYEE' GOTO EMPLOYEE;
3. -SALES
    TABLE FILE SALES
    .
    .
    .
    END
4. -EXIT
   -EMPLOYEE
    TABLE FILE EMPLOYEE
    .
    .
    .
    END
```

The procedure processes as follows:

1. The command -TYPE generates a message.
2. Assume the value passed to &PROC is SALES.
The -IF test checks the value of &PROC. Since it is not equal to EMPLOYEE, control passes to the label -SALES.
3. The FOCUS code is stacked. Control passes to the next line, -EXIT.
4. The command -EXIT executes the stacked commands. The output is sent to the terminal or output device and the procedure is exited.

The TABLE request under the label -EMPLOYEE is not executed.

This example also illustrates an *implicit exit*. If the value of &PROC was EMPLOYEE, control would pass to the label -EMPLOYEE after the -IF test, and the procedure would never encounter the -EXIT. The TABLE FILE EMPLOYEE request would execute and the procedure would automatically terminate.

Canceling Execution of the Procedure: -QUIT

-QUIT cancels execution of any stacked commands and causes an immediate exit from the procedure.

This command is useful if tests or computations generate results that make additional processing unnecessary.

Example

Canceling Execution of the Procedure

The following example illustrates the use of -QUIT to cancel execution based on the results of an -IF test.

```
1. -TYPE This report calculates percentage of returns.  
   TABLE FILE SALES  
   .  
   .  
   .  
   END  
2. -IF &CODE GT 'B10' OR &CODE EQ 'DONE' GOTO QUIT;  
3. -QUIT
```

The procedure processes as follows:

1. The command -TYPE generates a message. The FOCUS code is stacked.
2. Assume that the value of &CODE is B11.
The command -IF tests the value and passes control to -QUIT.
3. The command -QUIT cancels execution of the stacked commands and exits the procedure.

Exiting FOCUS and Setting Return Codes: -QUIT FOCUS

The Dialogue Manager command -QUIT FOCUS causes an immediate exit not only from the procedure, but from FOCUS as well. It returns you to the operating system and sets a return code.

Syntax

How to Exit FOCUS and Set a Return Code

```
-QUIT FOCUS [n|8]
```

where:

n|8

Is the operating system return code number. It can be a constant or variable. A variable should be an integer. If you do not supply a value or if you supply a non-integer value, the return code posted to the operating system is 8 (the default value).

A major function of user-controlled return codes is to detect processing problems. The return code value determines whether to continue or terminate processing. This is particularly useful for batch processing.

Branching

The execution flow of a procedure is determined with the following commands:

- -GOTO. Used for unconditional branching, -GOTO transfers control to a label.
- -IF...GOTO. Used for conditional branching, -IF...GOTO transfers control to a label depending on the outcome of a test.

-GOTO Processing

Dialogue Manager processes a -GOTO as follows:

- It searches forward through the procedure for the target label. If it reaches the end without finding the label, it continues the search from the beginning of the procedure.
- The first time through a procedure, Dialogue Manager notes the addresses of all the labels so that they can be found immediately if needed again.
- Dialogue Manager takes no action on labels that do not have a corresponding -GOTO.
- If a -GOTO does not have a corresponding label, execution halts and an error message is displayed.

Syntax

How to Unconditionally Branch With -GOTO

```
-GOTO label  
.  
.  
.  
-label [TYPE text]
```

where:

label

Is a user-defined name of up to 12 characters. Do not use embedded blanks or the name of any other Dialogue Manager command except -QUIT or -EXIT. Do not use words that can be confused with functions or arithmetic or logical operations.

The label may precede or follow the -GOTO command in the procedure.

TYPE text

Optionally sends a message to a client application.

Example

Unconditional Branching With -GOTO

The following example “comments out” all the FOCUS code using an unconditional branch rather than -* in front of every line:

```
-START TYPE PROCESSING BEGINS  
-GOTO DONE  
TABLE FILE SALES  
PRINT UNIT_SOLD RETURNS  
WHERE PROD_CODE BETWEEN '&CODE1' AND '&CODE2'  
AND PRODUCT = '&PRODUCT'  
BY PROD_CODE,CITY  
END  
-RUN  
-DONE
```

Syntax

How to Conditionally Branch With -IF...GOTO

```
-IF expression [THEN] GOTO label1; [ELSE IF...;]  
                                [ELSE GOTO label2;]
```

where:

expression

Is a valid expression. Literals need not be enclosed in single quotation marks unless they contain embedded blanks or commas.

THEN

Is an optional keyword that increases readability of the command.

GOTO *label*

Is a user-defined name of up to 12 characters. Do not use embedded blanks or the name of any other Dialogue Manager command except -QUIT or -EXIT. Do not use words that can be confused with functions or arithmetic or logical operations.

The label may precede or follow the -IF command in the procedure.

ELSE IF

Optionally specifies a compound -IF test. See *Compound -IF Tests* on page 3-18.

ELSE GOTO

Optionally passes control to *label2* when the -IF test fails.

The command -IF must end with a semicolon (;) to signal that all logic has been specified. Continuation lines must begin with a hyphen (-).

Example

Conditional Branching With -IF...GOTO

In the following example, control passes to the label -PRODSALES if &OPTION is equal to S. Otherwise, control falls through to the label -PRODRETURNS, the line following the -IF test.

```
-IF &OPTION EQ 'S' GOTO PRODSALES;
-PRODRETURNS
    TABLE FILE SALES
        .
        .
        .
    END
-EXIT
-PRODSALES
    TABLE FILE SALES
        .
        .
        .
    END
-EXIT
```

The following command specifies both transfers explicitly:

```
-IF &OPTION EQ 'S' GOTO PRODSALES ELSE
- GOTO PRODRETURNS;
```

Notice that the continuation line begins with a hyphen (-).

Compound -IF Tests

You can use compound -IF tests provided each test specifies a target label.

Example

Using Compound -IF Tests

In the following example, if the value of &OPTION is neither R nor S, the procedure terminates (GOTO QUIT). The -QUIT serves both as a target label for the GOTO and as an executable command.

```
-IF &OPTION EQ 'R' THEN GOTO PRODRETURNS ELSE IF
- &OPTION EQ 'S' THEN GOTO PRODSALES ELSE
- GOTO QUIT;
    .
    .
    .
-QUIT
```

Using Operators and Functions in -IF Tests

Expressions in an -IF test can include all FOCUS arithmetic and logical operators, as well as available functions. For information on expressions, see the *Creating Reports* manual. See the *Using Functions* manual for details on functions.

Example

Testing System and Statistical Variables

You can use system and statistical variables in -IF tests.

In the following example, if data (&LINES) is retrieved with the request, then the procedure branches to the label -PRODSALES; otherwise, it terminates.

```
TABLE FILE SALES
.
.
.
-IF &LINES NE 0 GOTO PRODSALES;
-EXIT
-PRODSALES
.
.
.
```

Screening Values With -IF Tests

To ensure that a supplied value is valid in a procedure, you can test it for the following:

- Presence
- Length
- Type

For instance, you would not want to perform a numerical computation on a variable for which alphanumeric data has been supplied.

Syntax

How to Test for the Presence of a Value

```
-IF &name.EXIST [expression ]GOTO label...;
```

where:

&name

Is a user-supplied variable.

.EXIST

Indicates that you are testing for the presence of a value. If a value is not present, a zero (0) is passed to the expression. Otherwise, a non-zero value is passed.

expression

Is the remainder of a valid expression that uses &name.EXIST as an amper variable.

GOTO label

Specifies a label to branch to.

Example

Testing for the Presence of a Variable

In the following example, if no value is supplied, &OPTION.EXIST is equal to zero and control is passed to the label -CANTRUN. The procedure sends a message to the client application and then exits. If a value is supplied, control passes to the label -PRODSALES.

```
-IF &OPTION.EXIST GOTO PRODSALES ELSE GOTO CANTRUN;
.
.
.
-PRODSALES
    TABLE FILE SALES
        .
        .
        .
    END
-EXIT
-CANTRUN
-TEXT TOTAL REPORT CAN'T BE RUN WITHOUT AN OPTION.
-EXIT
```

Syntax

How to Test for the Length of a Value

```
-IF &name.LENGTH expression GOTO label...
```

where:

&name

Is a user-supplied variable.

.LENGTH

Indicates that you are testing for the length of a value. If a value is not present, a zero (0) is passed to the expression. Otherwise, the number of characters in the value is passed.

expression

Is the remainder of a valid expression after *&name* is expanded.

GOTO label

Specifies a label to branch to.

Example

Testing for Variable Length

In the following example, if the length of &OPTION is greater than one, control passes to the label -FORMAT, which informs the client application that only a single character is allowed.

```
-IF &OPTION.LENGTH GT 1 GOTO FORMAT ELSE
-GOTO PRODSALES;
.
.
.
-PRODSALES
    TABLE FILE SALES
.
.
.
    END
-EXIT
-FORMAT
-TYPE ONLY A SINGLE CHARACTER IS ALLOWED.
```

Example

Storing the Length of a Variable

The following example sets the variable &WORDLEN to the length of the string contained in the variable &WORD:

```
-PROMPT &WORD. ENTER WORD.
-SET &WORDLEN = &WORD.LENGTH;
```

Syntax

How to Test for the Type of a Value

```
-IF &name.TYPE expression GOTO label...;
```

where:

&name

Is a user-supplied variable.

TYPE

Indicates that you are testing for the type of a value. The letter N (numeric) is passed to the expression if the value can be interpreted as a number up to 10^9-1 and can be stored in four bytes as a floating point format. In Dialogue Manager, the result of an arithmetic operation with numeric fields is truncated to an integer after the whole result of an expression is calculated. If the value could not be interpreted as numeric, the letter A (alphanumeric) or the letter U (undefined) is passed to the expression.

expression

Is the remainder of a valid expression after *&name* is expanded.

GOTO label

Specifies a label to branch to.

Example

Testing for Variable Type

In the following example, if &OPTION is not alphanumeric, control passes to the label -NOALPHA, which informs the client application that only alphanumeric characters are allowed.

```
-IF &OPTION.TYPE NE A GOTO NOALPHA ELSE
- GOTO PRODSALES;
.
.
.
-PRODSALES
  TABLE FILE SALES
  .
  .
  .
  END
-EXIT
-NOALPHA
-TYPE ENTER A LETTER ONLY.
```

Testing the Status of a Query

The system variable &RETCODE returns a code after a query is executed. If the query results in a normal display, the value of &RETCODE is 0. If a display error occurs, or no display results (as can happen when the query finds no data), the value of &RETCODE is 8. (If the error occurs on a ? SU, the value of &RETCODE is 16.)

The value of &RETCODE is set following the execution of any of these queries:

	NORMAL	NODISPLAY	ERROR
? HOLD	0	8	
? SU*	0	8	16
? JOIN	0	8	
? COMBINE	0	8	
? DEFINE	0	8	
? USE	0	8	
? LOAD	0	8	
? FILEDEF	0	8	

*The &RETCODE value of ? SU means: 0 indicates that the FOCUS Database Server (formerly called the sink machine) is up with one or more users; 8 indicates that the FOCUS Database Server is up with no users; 16 indicates that there is an error in communicating to the FOCUS Database Server.

You can test the status of any of these queries by checking the &RETCODE variable and providing branching instructions in your procedure.

For example, if you are using Simultaneous Usage (SU), you must know if the FOCUS Database Server is available before you can begin a particular procedure. The following procedure tests whether SINK1 is available before launching PROC1.

```
? SU SINK1
-RUN
-IF &RETCODE EQ 16 GOTO BAD;
-INCLUDE PROC1
-BAD
-EXIT
```

Looping

The Dialogue Manager command `-REPEAT` allows looping in a procedure.

Syntax

How to Specify a Loop

```
-REPEAT label                n TIMES  
-REPEAT label                WHILE condition  
-REPEAT label                FOR &variable [FROM fromval] [TO toval] [STEP s]
```

where:

label

Identifies the end of the code to be repeated (the loop). A label can include another loop if the label for the second loop has a different name from the first.

n TIMES

Specifies the number of times to execute the loop. The value of *n* can be a local variable, a global variable, or a constant. If it is a variable, it is evaluated only once, so the only way to end the loop early is with `-QUIT` or `-EXIT` (you cannot change the number of times to execute the loop) or to branch out of the loop.

WHILE *condition*

Specifies the condition under which to execute the loop. The condition is any logical expression that can be true or false. The loop is run if the condition is true.

FOR &*variable*

Is a variable that is tested at the start of each execution of the loop. It is compared with the value of *fromval* and *toval* (if supplied). The loop is executed only if &*variable* is less than or equal to *toval* (STEP is positive), or greater than or equal to *toval* (STEP is negative).

FROM *fromval*

Is a constant that is compared with &*variable* at the start of each execution of the loop. The default value is 1.

TO *toval*

Is a value that is compared with &*variable* at the start of each execution of the loop. The default is 1,000,000.

STEP *s*

Is a constant used to increment &*variable* at the end of each execution of the loop. It may be positive or negative. The default value is 1.

The parameters FROM, TO, and STEP can appear in any order.

Example

Using -REPEAT to Create a Loop

These examples illustrate how to write each of the syntactical elements of -REPEAT.

1. -REPEAT *label* *n* TIMES

Example:

```
-REPEAT LAB1 2 TIMES
-TYPE INSIDE
-LAB1 TYPE OUTSIDE
```

The output is:

```
INSIDE
INSIDE
OUTSIDE
```

2. -REPEAT *label* WHILE *condition*

Example:

```
-SET &A = 1;
-REPEAT LABEL WHILE &A LE 2;
-TYPE &A
-SET &A = &A + 1;
-LABEL TYPE END: &A
```

The output is:

```
1
2
END: 3
```

3. -REPEAT *label* FOR *&variable* FROM *fromval* TO *toval* STEP *s*

Example:

```
-REPEAT LABEL FOR &A STEP 2 TO 4
-TYPE INSIDE &A
-LABEL TYPE OUTSIDE &A
```

The output is:

```
INSIDE 1
INSIDE 3
OUTSIDE 5
```

Ending a Loop

A loop can end in one of three ways:

- It executes in its entirety.
- A -QUIT or -EXIT is issued.
- A -GOTO is issued to a label outside of the loop.

Note: If you later issue another -GOTO to return to the loop, the loop proceeds from the point it left off.

Using Expressions: -SET

The Dialogue Manager command -SET can be used in various ways to define values for variables in Dialogue Manager. You can compute new variables or recompute existing ones using arithmetic and logical expressions. You can also control loops, set indexes for variables, and call functions.

The following is a list of what can be included in a -SET expression and some specific rules for computations when using amper variables. Some calculations and special functions require that the amper variables have numeric values. FOCUS substitutes the value before placing the calculation in the stack. The variable does not have to have an I (integer) format, but the value for the variable must not contain alphanumeric characters. Note that the LAST operator used for reporting has no meaning in Dialogue Manager, nor do special MODIFY functions like FIND or LOOKUP.

- You can perform concatenations with the concatenation symbol. You must insert a space separating the amper variable from the concatenation symbol.
- You can use the DECODE function.
- You can use the EDIT function; however, its use is limited to the mask option.
- You can use the TRUNCATE function
- You can use the date functions.
- You can use other functions.

For more information on expressions, see the *Creating Reports* manual. For information on functions, see the *Using Functions* manual.

Computing a New Variable

You can use -SET to define a value for a substituted variable based on the results of a logical or arithmetic expression or a combination.

Syntax

How to Compute a New Variable

```
-SET &name = expression;
```

where:

&name

Is a user-supplied variable that has its value assigned with the expression.

expression

Is an expression following the rules outlined in the *Creating Reports* manual, but with limitations as defined in this topic. The semicolon after the expression is required to terminate the -SET command.

Example

Altering a Variable Value

The following example demonstrates the use of -SET to alter variable values based on tests.

```
-START
-TYPE RETAIL PRICE ABOVE OR BELOW $1.00 IN THIS REPORT?
-PROMPT &CHOICE. ENTER A OR B.
-SET &REL = IF &CHOICE EQ A THEN 'GT' ELSE 'LT';
  TABLE FILE SALES
  PRINT PROD_CODE UNIT_SOLD RETAIL_PRICE
  BY STORE_CODE BY DATE
  IF RETAIL_PRICE &REL 1.00
  END
```

In the example, the &CHOICE variable receives either A or B as the value supplied through -PROMPT. Assuming the user enters the letter A, -SET assigns the string value GT to &REL. Then, the value GT is passed to the &REL variable in the procedure, so that the expanded FOCUS command at execution time is:

```
IF RETAIL_PRICE GT 1.00
```

Note that literals are enclosed by single quotation marks. These are optional unless the literal contains embedded commas or blanks. To produce a literal that includes a single quotation mark, place two single quotation marks where you want one to appear.

Using the DECODE Function

You can use the DECODE function to change a variable to an associated value.

Example

Assigning a Value to a Variable With DECODE

In the following example, the variable refers to a label:

```
1.  -PROMPT &SELECT. ENTER CHOICE (A,B,C,D,E).
2.  -SET &GO=DECODE &SELECT (A ONE B TWO C THREE
    -D FOUR E FIVE ELSE EXIT);
3.  -GOTO &GO
    -ONE
    .
    .
    .
    -TWO
    .
    .
    .
```

The example processes as follows:

1. -PROMPT prompts the user at the terminal for a value for the variable &SELECT. Assume the user enters A.
2. -SET defines the variable &GO in terms of the DECODE function. Depending on the value input for &SELECT, DECODE associates a substitution. In this case, ONE is substituted for A.
3. -GOTO &GO transfers control to the label -ONE.

In the example, &GO can be another procedure (see *Incorporating Multiple Procedures* on page 3-37) that is executed, depending on the value that is decoded:

```
-TOP
-TYPE
-PROMPT &SELECT. ENTER 1, 2, 3, 4, 5, OR EXIT TO END.
-SET &GO=DECODE &SELECT (1 ONE 2 TWO 3 THREE
- 4 FOUR 5 FIVE ELSE EXIT);
-IF &GO IS EXIT GOTO EXIT;
EX &GO
-RUN
-GOTO TOP
-EXIT
```

For more information on DECODE, see the *Using Functions* manual.

Using the EDIT Function

You can use the mask option of the EDIT function with amper variables. You can insert characters into an alphanumeric value, or extract certain characters from the value.

Example

Using the EDIT Function With Amper Variables

In the following example, EDIT extracts a particular character, in this case the J, for comparison in order to branch to the appropriate label. Assume there are nested menus and the user must supply a number to branch to a particular menu. If the first character is a J, the branch is to the label JUMP that enables the user to jump in nested menus (the numbers refer to the explanation below):

```
1.  -TYPE CHOOSE 1 for Edit, 2 for Print, 3 for Math
1.  -TYPE TO JUMP LEVELS OF MENUS TYPE J1.3 ETC.
2.  -PROMPT &OPTION.A4.Please enter selection:.
3.  -SET &XYZ = EDIT(&OPTION, '9$$$');
4.  -IF &XYZ EQ J THEN GOTO JUMP;
    .
    .
    .
5.  -JUMP
    .
    .
    .
```

The example processes as follows:

1. -TYPE send messages to the screen explaining the options to the user.
2. -PROMPT asks the user to enter a value for the variable &OPTION. It can have as many as four characters.
3. -SET calculates the variable &XYZ, which is the &OPTION variable, using the mask option of EDIT. The first character is screened.
4. -IF determines the branch. If the variable &XYZ is equal to J, processing continues to the label JUMP. Otherwise, processing continues to the next command in the procedure.
5. -JUMP is a label. The coding that follows contains the necessary FOCUS commands to enable the user to jump to the various menus.

Using the TRUNCATE Function

The Dialogue Manager TRUNCATE function removes trailing blanks from Dialogue Manager amper variables and adjusts the length accordingly.

The Dialogue Manager TRUNCATE function has only one argument, the string or variable to be truncated. If you attempt to use the Dialogue Manager TRUNCATE function with more than one argument, the following error message is generated:

```
(FOC03665) Error loading external function 'TRUNCATE'
```

This function can only be used in Dialogue Manager commands that support function calls, such as -SET and -IF commands. It cannot be used in -TYPE or -CRTFORM commands or in arguments passed to stored procedures.

Note: A user-written function of the same name can exist without conflict.

Syntax

How to Use the TRUNCATE Function

```
-SET &var2 = TRUNCATE(&var1);
```

where:

&var2

Is the Dialogue Manager variable to which the truncated string is returned. The length of this variable is the length of the original string or variable minus the trailing blanks. However, if the original string consisted of only blanks, a single blank, with a length of one is returned.

&var1

Is a Dialogue Manager variable or a literal string enclosed in single quotation marks. System variables and statistical variables are allowed as well as user-created local and global variables.

Example

Using the Dialogue Manager TRUNCATE Function

The following example shows the result of truncating trailing blanks:

```
-SET &LONG = 'ABC   ' ;  
-SET &RESULT = TRUNCATE(&LONG);  
-SET &LL = &LONG.LENGTH;  
-SET &RL = &RESULT.LENGTH;  
-TYPE LONG   = &LONG   LENGTH = &LL  
-TYPE RESULT = &RESULT LENGTH = &RL
```

The output is:

```
LONG   = ABC      LENGTH = 06  
RESULT = ABC      LENGTH = 03
```

The following example shows the result of truncating a string that consists of all blanks:

```
-SET &LONG = '           ' ;  
-SET &RESULT = TRUNCATE(&LONG);  
-SET &LL = &LONG.LENGTH;  
-SET &RL = &RESULT.LENGTH;  
-TYPE LONG   = &LONG   LENGTH = &LL  
-TYPE RESULT = &RESULT LENGTH = &RL
```

The output is:

```
LONG   =           LENGTH = 06  
RESULT =           LENGTH = 01
```

The following example uses the TRUNCATE function as an argument for EDIT:

```
-SET &LONG = 'ABC   ' ;  
-SET &RESULT = EDIT(TRUNCATE(&LONG) | 'Z', '9999');  
-SET &LL = &LONG.LENGTH;  
-SET &RL = &RESULT.LENGTH;  
-TYPE LONG   = &LONG   LENGTH = &LL  
-TYPE RESULT = &RESULT LENGTH = &RL
```

The output is:

```
LONG   = ABC      LENGTH = 06  
RESULT = ABCZ     LENGTH = 04
```

Controlling a Loop With -SET

You can use the -SET command to control the repetition limit of a loop.

Example

Controlling a Loop With -SET

In the following example, the variable &N is incremented using -SET and tested to terminate the loop:

```
1.  -DEFAULTS &N=0
2.  -START
3.  -SET &N=&N+1;
4.      EX SLRPT
    -RUN
5.  -IF &N GT 5 GOTO NOMORE;
6.  -GOTO START
5.  -NOMORE TYPE EXCEEDING REPETITION LIMIT
    -EXIT
```

Execution proceeds in this way:

1. The -DEFAULTS command initializes the loop-controlling variable &N to 0.
2. -START is a Dialogue Manager label that begins the loop. It is the target of an unconditional -GOTO.
3. The -SET command increments the value of &N by one each time through the loop.
4. The FOCUS command EX SLRPT is stacked. The command -RUN then calls for the execution of the stacked command.
5. This -IF command tests the current value of the variable &N. If the value is greater than 5, control passes to the label -NOMORE, which displays a message for the user and forces an exit. If the value of &N is 5 or less, control falls through to the next Dialogue Manager command.
6. The unconditional Dialogue Manager command -GOTO START causes the loop to repeat.

Setting a Date

Natural date literals can be used in Dialogue Manager. They provide a way to take advantage of the powerful date handling capabilities of FOCUS.

Example

Setting Dates and Computing the Difference in Days

Consider the following example:

```
-SET &NOW= 'MAR 11 1999';  
-SET &LATER= '2000 11 MAR';  
-SET &DELAY = &LATER - &NOW;
```

The value of &DELAY is set to the difference, in days, between &LATER and &NOW.

Note:

- A computation that adds or subtracts a fixed number of days from a variable in DATE format is not yet supported.
- A date given to Dialogue Manager cannot exceed 20 characters, including spaces.
- Dialogue Manager accepts only full-format dates (that is, MDY or MDYY, in any order).

Calling a Function

Any function name encountered in a Dialogue Manager expression that is not recognized as a system standard name or FOCUS function is assumed to be a function. These functions are externally programmed by users and stored in a library that is available at the time they are referenced. One or more arguments are passed to the user program, which performs an operation or calculation and returns a single value or character string.

Dialogue Manager variables can receive their values from functions through -SET.

Syntax

How to Set a Variable Value Based on the Result From a Function

```
-SET &name = routine(argument,...,'format');
```

where:

name

Is the name of the variable in which the result is stored.

routine

Is the name of the function.

argument

Represents the argument(s) that must be passed to the function. These arguments are converted to decimal format.

format

Is the predefined format of the result. This is used to convert the numeric format back to character representation. It must be enclosed in single quotation marks.

Example

Setting a Variable Value Based on the Result From a Function

In the following example, FOCUS invokes the function RATE, adds 0.5 to the calculated value, and then formats the result as a double precision number. This result is then stored in the variable &COST:

```
-PROMPT &COMPANY.WHAT COMPANY ARE YOU USING?.  
-PROMPT &DEST.WHERE ARE YOU SENDING THE PACKAGE TO?.  
-PROMPT &WEIGHT.HOW HEAVY IS THE PACKAGE IN POUNDS?.  
-SET &COST = RATE(&COMPANY,&DEST,&WEIGHT,'D6.2') + 0.5;  
-TYPE THE COST TO SEND A &WEIGHT pound PACKAGE  
-TYPE TO &DEST BY &COMPANY IS &COST
```

Syntax

How to Load and Execute a Function

The following is an alternate way of calling functions. The Dialogue Manager command causes the function to be loaded and then executed. The syntax is

```
{-CMS } RUN routine, argument,...  
{-TSO } RUN routine, argument,...  
{-MVS } RUN routine, argument,...
```

where:

routine

Is the name of the function.

argument

Represents the argument(s) that must be passed to the function. Arguments that are variables must have sizes predefined in prior -SET commands.

The numeric arguments to the function are not automatically converted to D format in this syntax. Any required conversion must be done externally by the user or in the function.

Example

Loading and Executing a Function

```
-PROMPT &MYCODE.A3.  
-SET &MYNAME = '' ;  
-SET &MYFACTOR = '' ;  
-CMS RUN CODENAME,&MYCODE,&MYNAME,&MYFACTOR
```

In this example the program is CODENAME. The arguments that are variables are either prompted for or set at the beginning of the procedure and the values are then supplied for the arguments. Note that in this syntax the user program may use an argument for both input and output purposes. It is the responsibility of the user program to move the correct number of characters into the output variables.

Additional Facilities

Dialogue Manager supports a number of facilities for building applications. These facilities include:

- Creating startup files (profiles) that set overall environment conditions, which apply throughout your working session with FOCUS.
- Using -INCLUDE and EXEC to dynamically insert a procedure in another procedure, or to nest them up to four levels.
- Creating windows and menus for displaying information and collecting data in a procedure.
- Debugging procedures.
- Managing data integrity and security.
- Transferring data to and from non-FOCUS files.

Establishing Startup Conditions

FOCUS supports a startup profile that executes its content immediately upon entry into FOCUS. Using this procedure you can:

- Establish standard conditions that apply throughout the subsequent working session. For example, you can predefine environment parameters or automatically compute variables and make them available for later use.
- Provide a menu of subsequent user options.
- Control use of an application.

You can create a profile using any text editor or the FOCUS editor TED. The file is a FOCEXEC named PROFILE.

Note: It is possible to use an alternate FOCEXEC as a profile or not to execute a profile at all. For more information, see the *Overview and Operating Environments* manual.

Example Creating a Startup Profile

Note the following example of a profile (under CMS):

```
USE
SALES FOCUS A1
MASTER FOCUS C1
END
CMS FILEDEF MYSAV DISK SAVE TEMP (LRECL 304 RECFM V
DEFINE FILE SALES
RATIO/D5.2 = (RETURNS/UNIT_SOLD);
END
-TYPE FOCUS SESSION ON &DATE MDYY &TOD

LET WORKREPORT=TABLE FILE EMPLOYEE
SET LINES=57, PAPER=66, PAGE=OFF
OFFLINE
```

Upon entering FOCUS, the profile is executed and a message showing the current date and time is displayed:

```
FOCUS SESSION ON 03/11/99 AT 14:21:06
```

Incorporating Multiple Procedures

Dialogue Manager supports dynamic inclusion of other procedures into a stored procedure at run time to enhance efficiency. There are two ways to do this:

- You can use the EXEC command in a procedure. The command will be stacked with other FOCUS commands and executed when an appropriate Dialogue Manager command forces execution of the stack. The procedure must be a fully executable procedure.
- The -INCLUDE command incorporates a file, which may be whole or partial procedures. A partial procedure could not be executed alone, but can be saved in a file and included in a calling procedure. This is particularly useful for procedures containing common header text, or partial processing cases that can be included at run time, based on tests and branches initiated in the original procedure. The level of nesting is limited only by the available memory.

The major difference between these two methods is when the procedure is executed. An EXEC command would be stacked and subsequently executed when the appropriate Dialogue Manager command is encountered, whereas -INCLUDE occurs immediately.

You cannot perform recursive nesting of -INCLUDE or EX commands. That is, a file that is inserted using the -INCLUDE command cannot directly or indirectly -INCLUDE or EX itself. An indirect recursion would occur if procedure A contained a -INCLUDE command for procedure B and procedure B contained a -INCLUDE or EX command for procedure A.

Using -INCLUDE

Lines inserted from an -INCLUDE are incorporated into the calling procedure as if they had originally been placed there.

There are many more uses for -INCLUDE files:

- As a control over the user environment. The included procedure must be present and some switches set before the present procedure continues execution.
- As a security mechanism. The included procedure can be encrypted and a direct password set. For more information, see the *Describing Data* manual.
- The name of the included file can be determined by the procedure (for example, -INCLUDE &NEWLINES, where NEWLINES is a variable whose value is a file name). This can shorten the main procedure when there are many alternate procedures.

Syntax

How to Incorporate a File

```
-INCLUDE filename [filetype [filemode]]
```

where:

filename

Is the name of a FOCUS procedure.

filetype

Is the procedure's file type on CMS. If none is included, a file type of FOCEXEC is assumed.

filemode

Is the procedure's file mode on CMS. If none is included, a file mode of A is assumed.

Example**Incorporating a File**

```

-IF &OPTION EQ S GOTO PRODSALES
-ELSE GOTO PRODRETURNS;
.
.
.
-PRODRETURNS
-INCLUDE DATERPT
-RUN
.
.
.

```

Assume that DATERPT is a procedure containing the following TABLE request:

```

TABLE FILE SALES
PRINT PROD_CODE UNIT_SOLD
BY STORE_CODE
IF PROD_CODE IS &PRODUCT
END

```

-INCLUDE incorporates this request into the calling procedure. FOCUS prompts for a value for the variable &PRODUCT as soon as the -INCLUDE is encountered. The ensuing -RUN forces the execution of this included TABLE request.

Example**Incorporating Non-Executable Code**

You can use -INCLUDE to call files containing code that is not executable. For instance, a common heading used throughout all reports can be stored in a separate file and incorporated into any procedure as needed. For example,

```

TABLE FILE SALES
-INCLUDE SALEHEAD
SUM UNIT_SOLD AND RETURNS AND COMPUTE ...

```

where the SALEHEAD file contains:

```

HEADING
"THE ABC CORPORATION"
"RETAIL SALES DIVISION"
"MONTHLY SALES REPORT"

```

Example Incorporating a Virtual Field

A virtual field can be placed in a separate file and called from a procedure as follows

```
-INCLUDE DEFRATIO
TABLE FILE SALES
-INCLUDE SALEHEAD
SUM UNIT_SOLD AND RETURNS AND RATIO
BY CITY
.
.
.
```

where the DEFRATIO file contains:

```
DEFINE FILE SALES
RATIO/D5.2=(RETURNS/UNIT_SOLD);
END
```

This DEFINE will be dynamically included before the TABLE request executes.

Nesting Procedures With -INCLUDE

Any number of different procedures can be invoked from a single calling procedure. You can also nest -INCLUDE commands within each other. The level of nesting is limited only by the available memory.

```
-PRODSALES
-INCLUDE FILE1
-RUN
```

```
FILE1
-INCLUDE FILE2
-RUN
```

```
FILE2
-INCLUDE FILE3
-RUN
```

```
FILE3
-INCLUDE FILE4
-RUN
```

```
FILE4
-RUN
```

Files 1 through 4 are incorporated into the original procedure. All of the included files are viewed as part of the original procedure. A procedure cannot branch to a label in an included file or recursively include a file.

Using EXEC

A procedure can also call another one with the command EXEC (EX). The called procedure must be fully executable. You can also pass values to variables on the command line.

Example

Using EXEC to Call a Procedure

In the following example, a procedure calls DATERPT:

```
-IF &OPTION EQ 'S' GOTO PRODSALES ELSE GOTO PRODRETURNS;  
.  
.  
.  
-PRODRETURNS  
  EX DATERPT  
.  
.  
.  
-RUN
```

Note: If the last executable command in the called procedure is a -CRTFORM, control will not be returned to the calling procedure unless another Dialogue Manager command is included to terminate the -CRTFORM, such as -RUN or a -label.

Developing an Open-Ended Procedure

A file of stored FOCUS commands without variables looks and executes exactly as though it had been typed interactively into FOCUS from the terminal. However, if there is an error in your procedure file, it is rejected. If you make an error while typing interactively from the terminal, FOCUS issues prompts to help you correct the error.

If you store a procedure without the END command, you can execute all the procedure lines. The terminal then opens to allow interactive completion of the procedure. You can add additional command lines and enter the END command from the terminal to complete the procedure.

Note that you cannot use amper variables when typing online at a terminal. Open-ended procedures do not support variable substitution in lines entered after the terminal is opened. Variable substitution is supported in the stored portion of the procedure.

Example Developing and Running an Open-Ended Procedure

Assume the following open-ended procedure is stored as SLRPT:

```
-TYPE ENTER REST OF PROCEDURE
TABLE FILE SALES
HEADING CENTER
"MONTHLY REPORT"
SUM UNIT_SOLD AND RETURNS AND COMPUTE
RATIO/D5.2 = 100 * RETURNS/UNIT_SOLD;
```

You can invoke the procedure by typing EX SLRPT. It executes normally but fails to encounter an END command in the file. It then opens up the terminal displaying the FOCUS prompt. Depending on what you want, you could supply:

```
BY STORE_CODE
END
```

Or, alternatively:

```
IF CITY IS STAMFORD
BY STORE_CODE
END
```

Debugging With &ECHO

It can be helpful to display command lines as they execute in order to test and debug procedures. The variable &ECHO is available for this purpose.

Syntax How to Display Command Lines as They Execute

```
&ECHO = display
```

Valid values are:

ON

Displays lines that are expanded and stacked for execution.

ALL

Displays Dialogue Manager commands as well as lines that are expanded and stacked for execution.

OFF

Suppresses display of both stacked lines and Dialogue Manager commands. This value is the default.

You can set &ECHO through -DEFAULTS, -SET, or on the command line. For example, you can set ECHO to ALL for the execution of the procedure SLRPT using any of the following commands:

```
-DEFAULTS &ECHO = ALL
```

or

```
-SET &ECHO = ALL;
```

or

```
EX SLRPT ECHO = ON
```

If you use -SET or -DEFAULTS and place it in the procedure, display begins from that point in the procedure, and can be turned off and on again at any other point in the procedure.

Note that if the procedure is encrypted, &ECHO automatically receives the value OFF, regardless of the value that is assigned explicitly.

Testing Dialogue Manager Command Logic With &STACK

To test the logic of Dialogue Manager commands, you can run the procedure but prevent actual execution of the stacked commands by setting the &STACK variable.

Syntax

How to Test Dialogue Manager Command Logic

```
&STACK = {ON|OFF}
```

where:

ON

Results in normal execution of stacked commands. This value is the default.

OFF

Prevents execution of stacked commands. In addition, system variables (for example, &RECORDS or &LINES) are not set. Dialogue Manager commands are executed so you can test the logic of the procedure.

You can set &STACK through -DEFAULTS, -SET, or on the command line. For example, you can set &STACK to OFF for the execution of the procedure SLRPT using any of the following commands:

```
-DEFAULTS &STACK = OFF
```

or

```
-SET &STACK = OFF;
```

or

```
EX SLRPT STACK = OFF
```

This is usually used with ECHO = ALL for debugging purposes. The terminal displays both the Dialogue Manager commands, as well as the FOCUS commands with the supplied values. You can view the logic of the procedure.

Locking Procedure Users Out of FOCUS

Normally, users can respond to a Dialogue Manager value request with QUIT and return to the FOCUS command level or the prior procedure. In situations where it is important to prevent users from entering native FOCUS or QUIT from a particular procedure, the environment can be locked and QUIT deactivated.

Syntax

How to Lock Procedure Users Out of FOCUS

Enter the following command within the procedure:

```
-SET &QUIT=OFF;
```

With QUIT deactivated, any attempt to return to native FOCUS produces an error message indicating that “quit” is not a valid value. Then the user is prompted for another value.

A user can terminate the FOCUS session from inside a locked procedure by responding to a prompt with

```
QUIT FOCUS
```

to return to the operating system, not the FOCUS command level.

Note: The default value for &QUIT is ON.

Writing to Files: -WRITE

In addition to conducting a dialogue with the user, Dialogue Manager can read from and write to files. For information on reading values from files, see *Supplying Values Without Prompting* on page 3-68.

The Dialogue Manager -WRITE enables you to write lines of text to a file.

Syntax

How to Write to a File

```
-WRITE ddname [NOCLOSE] text
```

where:

ddname

Is the logical name of the file as defined to FOCUS using FILEDEF, ALLOCATE, or DYNAM ALLOCATE. For information about file allocations, see the *Overview and Operating Environments* manual.

NOCLOSE

Indicates that the file should be kept open even if a -RUN is encountered. The file is closed upon completion of the procedure or when a -CLOSE or subsequent -READ command is encountered.

text

Is any combination of variables and text. To write more than one line, end the first line with a comma (,) and begin the next line with a hyphen followed by a space (-).

-WRITE opens the file to receiving the text and closes it upon exit from the procedure. When the file is reopened for writing, the new material overwrites the old. If you wish to reopen to add new records instead of overwriting existing ones, use the attribute DISP MOD when you define the file to the operating system.

Example

Writing to a File

The following example reopens the file PASS under CMS to add new text:

```
-CMS FILEDEF PASS DISK PASS DATA (DISP MOD
-WRITE PASS &DIV &RED &TEST RESULT IS,
- &RECORDS AT END OF RUN
```

Example Reading From and Writing to Sequential Files

The following example illustrates reading from and writing to sequential files and the use of operating system commands (in this example, CMS). The numbers in the margin refer to notes that follow the example.

```
1.  -TOP
2.  -PROMPT &CITY. ENTER NAME OF CITY -- TYPE QUIT WHEN DONE.
3.  -CMS FILEDEF PASS DISK PASS DATA A (LRECL 80 RECFM FB
4.  -WRITE PASS &CITY
      TABLE FILE SALES
      HEADING CENTER
      "LOWEST MONTHLY SALES FOR &CITY"
      " "
      PRINT DATE PROD_CODE
      BY LOWEST 1 UNIT_SOLD
      BY STORE_CODE
      BY CITY
      IF CITY EQ &CITY
      FOOTING CENTER
      "CALCULATED AS OF &DATE"
      ON TABLE SAVE AS INFO
      END
5.  -RUN
6.  -CMS FILEDEF LOG DISK LOG DATA A1 (LRECL 80 RECFM FB
      MODIFY FILE SALES
      COMPUTE
      TODAY/I6=&YMD;
      CITY='&CITY';
      FIXFORM X5 STORE_CODE/A3 X15 DATE/A4 PROD_CODE/A3
      MATCH STORE_CODE DATE PROD_CODE
      ON MATCH TYPE ON LOG
      "<STORE_CODE><DATE><PROD_CODE><TODAY>"
      ON MATCH DELETE
      ON NOMATCH REJECT
      DATA ON INFO
      END
7.  -RUN
      EX SLRPT3
8.  -RUN
11. -GOTO TOP
12. -QUIT
```

The procedure SLRPT3, which is invoked from the calling procedure, contains the following lines:

```
9.  -READ PASS &CITY.A8.  
      TABLE FILE SALES  
      HEADING CENTER  
      "MONTHLY REPORT FOR &CITY"  
      "LOWEST SALES DELETED"  
      " "  
      PRINT PROD_CODE UNIT_SOLD RETURNS DAMAGED  
      BY STORE_CODE  
      BY CITY  
      IF CITY EQ &CITY  
      FOOTING CENTER  
      "CALCULATED AS OF &DATE"  
      END  
  
10. -RUN
```

The following paragraphs explain the logic and show the dialogue between the user and the screen. User entries are in lowercase:

1. -TOP marks the beginning of the procedure.
2. -PROMPT sends the following prompt to the screen after the procedure is executed:

```
ENTER NAME OF CITY -- TYPE QUIT WHEN DONE<stamford
```
3. FILEDEF defines and opens a file named PASS.
4. -WRITE writes the value of &CITY to the non-FOCUS file named PASS. In this case the value written is STAMFORD.

5. -RUN executes the stacked TABLE request. In this case, a non-FOCUS file named INFO is created with the SAVE command. This is a sequential file, containing the result of the TABLE request as shown below.

```
NUMBER OF RECORDS IN TABLE= 7 LINES= 7
(BEFORE TOTAL TESTS)

EBCDIC RECORD NAMED INFO

FIELDNAME      ALIAS      FORMAT      LENGTH
UNIT_SOLD      SOLD      I5          5
STORE_CODE     SNO       A3          3
CITY           CTY       A15         15
DATE           DTE       A4MD         4
PROD_CODE     PCODE     A3          3
-----
TOTAL                                     30

DEFAULT FILEDEF ISSUED

FILEDEF INFO DISK INFO FOCTEMP A1 (LRECL 30 BLKSIZE 300 RECFM F6)
>
>
```

6. FILEDEF defines a log file for the subsequent MODIFY request.
7. -RUN executes the stacked MODIFY request. The data comes directly from the INFO file created in the prior TABLE request and is entered using FIXFORM. Hence, the product with the lowest UNIT_SOLD is deleted from the file, and logged to a log file.

sales.foc ON 04/23/93 AT 12.28.38

```
TRANSACTIONS: TOTAL=      1  ACCEPTED=      1  REJECTED=      0
SEGMENTS:      INPUT=      0  UPDATED =      0  DELETED =      1
```

8. The next -RUN executes another procedure called SLRPT3.
9. -READ reads the value for &CITY from the non-FOCUS file PASS. In this case the value passed is STAMFORD.

10. The -RUN executes the TABLE request and control is routed back to the calling procedure.

PAGE 1

**MONTHLY REPORT FOR STAMFORD
LOWEST SALES DELETED**

STORE_CODE	CITY	PROD_CODE	UNIT_SOLD	RETURNS	DAMAGED
14B	STAMFORD	B10	60	10	6
		B12	40	3	3
		B17	29	2	1
		C7	45	5	4
		D12	27	0	0
		E2	80	9	4
		E3	70	8	9

CALCULATING AS OF 04/23/93

11. -GOTO TOP routes control to the top.
12. When the user types QUIT, processing ends.

Using Variables in Procedures

Interactive variable substitution is at the heart of Dialogue Manager. You can create procedures that include variables (also called amper variables) and supply values for them at run time. These variables store a string of text or numbers and can be placed anywhere in a procedure. A variable can refer to a field, a command, descriptive text, a file name—literally anything.

Variables can be used only in procedures. They are ignored if you use them while creating reports live at the terminal. Values for variables may be supplied either directly on the command line when you execute the procedure, or through the -DEFAULTS command, the -SET command, or a -READ command in the procedure itself.

This topic describes how to use amper variables in procedures and how to supply values for them. Variables fall into two classifications:

- Local and global variables have values supplied at run time. Local variable values remain in effect for the respective procedure, while global variable values remain in effect for all procedures executed during an entire FOCUS session (that is, from the time you enter FOCUS until you exit with the FIN command).
- System, statistical, and special variables have values that the system automatically resolves whenever you request them.

Leading double ampersands (&&) denote global variables. All other Dialogue Manager variables begin with a single ampersand (&). For this reason, in the FOCUS community they are known as amper variables.

The maximum number of local, global, system, statistical, special and index variables available in a procedure is 512. Approximately 30 are reserved for use by FOCUS.

Additionally, Dialogue Manager supports four types of prompting. You can alter the execution flow of your procedure, or change the substance of the request based on the values entered. These are

- **Direct Prompting with -PROMPT:** You can request a set of values before they are actually needed. You can write your own text for these prompts and then validate the entered values to confirm that they fit a preset list of acceptable items or match a predefined format.
- **Full-Screen Data Entry with -CRTFORM:** The -CRTFORM command gathers variable values through full-screen data entry. Many values can be input and manipulated at the same time. Several screens can be included in a single procedure and used for a variety of purposes, including the development of menu-driven applications.

-CRTFORM invokes FIDEL, the FOCUS Interactive Data Entry Language, and incorporates most of its functions. You can also use Screen Painter to design and paint -CRTFORM data entry screens directly on your terminal screen.

Note that the Dialogue Manager command -CRTFORM is used for entering Dialogue Manager amper variable values. The equivalent MODIFY command, CRTFORM (without a hyphen), is used in MODIFY requests to enter field values.

- **Selecting Items from a Menu with -WINDOW:** You can create a series of menus and windows using the Window Painter facility and display them on the screen using the -WINDOW command. When displayed, the menus and windows can collect data by prompting users to select a value, enter a value, or press a program function (PF) key.
- **Implied Prompting:** FOCUS recognizes variables in a procedure by the leading ampersand (&). If a value has not been provided by some other means, FOCUS automatically requests a value from the terminal when needed.

Querying the Values of Variables

Amper variable values can be queried during execution.

Syntax

How to Query the Values of Variables

```
-? &[string]
```

where:

string

Is a complete amper variable or a partial string of up to 12 characters. Only amper variables starting with the specified string are displayed.

The command displays the following message, followed by a list of currently defined amper variables and their values:

```
CURRENTLY DEFINED & VARIABLES:
```

Note that this is a Dialogue Manager query. Since local variables do not exist outside a procedure, no similar query is available from the FOCUS command line.

Querying Parameter Value Settings

There is a Dialogue Manager query that enables you to capture previously defined SET parameter values in amper variables.

Syntax

How to Query Parameter Value Settings

```
-? SET parameter ampervar
```

where:

parameter

Is any valid FOCUS setting that may be queried with the ? SET or ? SET ALL command.

ampervar

Is the name of the variable where the value is to be stored.

Example

Querying a Parameter Value Setting

For example, if you enter

```
-? SET ASNAMES &abc  
-TYPE &ABC
```

the value stored in &abc becomes the value of ASNAMES. If you omit &abc from the command, then a variable called &ASNAMES is created that contains the value of ASNAMES.

Local Variables

Local variables are identified by a single ampersand (&) preceding the name of the variable. They remain in effect throughout a single procedure.

Example

Using Local Variables

In the following example, &CITY, &CODE1, and &CODE2 are local variables:

```
TABLE FILE SALES
HEADING CENTER
"MONTHLY REPORT FOR &CITY"
"PRODUCT CODES FROM &CODE1 TO &CODE2"
" "
SUM UNIT_SOLD AND RETURNS AND COMPUTE
RATIO/D5.2 = 100 * (RETURNS/UNIT_SOLD);
BY CITY
IF CITY EQ &CITY
BY PROD_CODE
IF PROD_CODE IS-FROM &CODE1 TO &CODE2
END
```

Assume you supply the values when you execute the procedure:

```
EX SLRPT CITY = STAMFORD, CODE1=B10, CODE2=B20
```

The procedure looks like this before it processes:

```
TABLE FILE SALES
HEADING CENTER
"MONTHLY REPORT FOR STAMFORD"
"PRODUCT CODES FROM B10 TO B20"
" "
SUM UNIT_SOLD AND RETURNS AND COMPUTE
RATIO/D5.2 = 100 * (RETURNS/UNIT_SOLD);
BY CITY
IF CITY EQ STAMFORD
BY PROD_CODE
IF PROD_CODE IS-FROM B10 TO B20
END
```

Values supplied for local variables remain current in the procedure. That is, all instances of the variables receive the values supplied. However, the values are not passed to other procedures containing the same variables (that is, &CODE1 and &CODE2 in another procedure). The values disappear after SLRPT has finished processing.

Global Variables

Global variables differ from local variables in that once a value is supplied, it remains current throughout the FOCUS session, unless set to another value with -SET or cleared by the LET CLEAR command. For information on LET CLEAR, see Chapter 4, *Defining a Word Substitution*. They are useful for gathering values at the start of a work session for use by several subsequent procedures. All procedures that use a particular global variable receive the current value until you exit from FOCUS.

Global variables are specified through the use of a double ampersand (&&) preceding the variable name. It is possible to have a local and global variable with the same name. They are distinct and may have different values.

Example

Using Global Variables

The following is an example of a procedure containing global variables:

```
TABLE FILE SALES
HEADING CENTER
"MONTHLY REPORT FOR &&CITY"
SUM UNIT_SOLD AND RETURNS AND COMPUTE
RATIO/D5.2 = 100 * (RETURNS/UNIT_SOLD);
BY CITY
IF CITY EQ &&CITY
BY PROD_CODE
IF PROD_CODE IS-FROM &&CODE1 TO &&CODE2
END
```

Syntax

How to Query the Values of Global Variables

Since global variable values remain current throughout the FOCUS session, it is helpful to be able to display their values on demand. Do this by issuing the following command,

```
? &&
```

which displays the values of all global variables in use during the FOCUS session.

Example Querying the Values of Global Variables

The following example displays the values of three global variables:

```
>
? &&
  &&CITY          STAMFORD
  &&CODE1          B10
  &&CODE2          B20
>
```

System Variables

FOCUS automatically substitutes values for system variables encountered in a Dialogue Manager request. System-supplied variables cannot be overridden. For example, you can use the system variable &DATE to automatically incorporate the system date in your request.

Reference Summary of System Variables

A list of Dialogue Manager system variables follows:

Variable	Format or Value	Description
&DATE	MM/DD/YY	Returns the current date.
&DATEfmt	Any date format.	Returns the current date, where fmt can be any combination of YYMD, MDYY, etc.
&DMY	DDMMYY	Returns the current date.
&DMYY	DDMMCCYY	Returns the current (four-digit year) date.
&FOCCPU	milliseconds	Calculates the OS CPU time. MVS only. In CMS, this returns the same value as &FOCTTIME.
&FOEXTTRM	ON OFF	Indicates the availability of extended terminal attributes.
&FOCFIELDNAME	NEW OLD NOTRUNC	Returns a string indicating whether long and qualified field names are supported. A value of OLD means that they are not; NEW means that they are; and NOTRUNC means that they are supported, but unique truncations of field names cannot be used.

Variable	Format or Value	Description
&FOCFOCEXEC		Manages reporting operations involving many similarly named requests that are executed using EX. &FOCFOCEXEC enables you to easily determine which procedure is running. &FOCFOCEXEC can be specified within a request or in a Dialogue Manager command to display the name of the currently running procedure.
&FOCINCLUDE		Manages reporting operations involving many similarly named requests that are included using -INCLUDE. &FOCINCLUDE can be specified within a request or in a Dialogue Manager command to display the name of the current included procedure.
&FOCMODE	CMS CRJE MSO OS TSO	Identifies the operating environment.
&FOCPRINT	ONLINE OFFLINE	Returns the current print setting.
&FOCPUTLVL	FOCUS PUT level number .	(For example, 9306 or 9310.)&FOCPUTLVL is no longer supported.
&FOCQUALCHAR	. : ! % \	Returns the character used to separate the components of qualified field names.
&FOCREL	release number	Identifies the FOCUS Release number (for example, 6.5 or 6.8).
&FOCSBORDER	ON OFF	Whether solid borders will be used in full-screen mode.
&FOCSYSTYP	HIPER CP/A	CMS system type.

Variable	Format or Value	Description
&FOCTMPDSK	A ... Z	Identifies the disk where FOCUS places temporary work files (for example, HOLD files). CMS only.
&FOCTRMSD	24 27 32 43	Indicates terminal height. (This can be any value; the examples shown are common settings.)
&FOCTRMSW	80 132	Indicates terminal width. (This can be any value; the examples shown are common settings.)
&FOCTRMTYP	3270 TTY UNKNOWN	Identifies the terminal type.
&FOCTTIME	milliseconds	Calculates total CPU time. CMS only.
&FOCVTIME	milliseconds	Calculates virtual CPU time. CMS only.
&HIPERFOCUS	ON OFF	Returns a string showing whether HiperFOCUS is on.
&IORETURN		Returns the code set by the last Dialogue Manager -READ or -WRITE operation.
&MDY	MMDDYY	Returns the current date. The format makes this variable useful for numerical comparisons.
&MDYY	MMDDCCYY	Returns the current (four-digit year) date.
&RETCODE	numeric	Returns the return code set upon execution of an operating system command. Executes all FOCUS commands in the FOCSTACK just as the -RUN command would.
&TOD	HH.MM.SS	Returns the current time. When you enter FOCUS, this variable is updated to the current system time only when you execute a MODIFY, SCAN, or FSCAN command. To obtain the exact time during any process, use the HHMMSS function.
&YMD	YYMMDD	Returns the current date.
&YYMD	CCYYMMDD	Returns the current (four-digit year) date.

Example Using the System Variable &DATE

The following example illustrates the use of a system variable in a request:

```
TABLE FILE SALES
.
.
.
FOOTING "CALCULATED AS OF &DATEMDYY"
END
-EXIT
```

The system variable &DATEMDYY ensures that the date that appears in the report is always the current system date.

Example Using the System Variable &FOCFOCEXEC

This next example illustrates how to use the system variable &FOCFOCEXEC in a request to display the name of the currently running procedure:

```
TABLE FILE EMPLOYEE
"REPORT: &FOCFOCEXEC -- EMPLOYEE SALARIES"
PRINT CURR_SAL BY EMP_ID
END
```

If the request is stored as a procedure called SALPRINT, when executed it will produce the following:

```
REPORT: DA0219  -- EMPLOYEE SALARIES
EMP_ID          CURR_SAL
-----
071382660      $11,000.00
112847612      $13,200.00
115360218          $.00
117593129      $18,400.00
119265415       $9,500.00
119329144      $29,700.00
121495681          $.00
123764317      $26,862.00
126724188      $21,120.00
219984371      $10,400.00
326179357      $21,700.00
451123478      $16,100.00
543729165       $9,000.00
818692173      $27,062.00
```

&FOCFOCEXEC and &FOCINCLUDE can also be used in -TYPE commands. For example, you have a procedure named EMPNAME that contains the following:

```
-TYPE &|FOCFOCEXEC is: &FOCFOCEXEC
```

When EMPNAME is executed, the following output is produced:

```
&FOCFOCEXEC IS: EMPNAME
```

Displaying a Date Variable Containing a Four-Digit Year

You can display a date variable containing a 4-digit year without separators. The variables are &YYMD, &MDYY, and &DMYY. These variables complement the 2-digit year variables &YMD, &MDY, and &DMY.

Example

Using the System Variable &YYMD

The following example shows a report using &YYMD:

```
TABLE FILE EMPLOYEE
HEADING
"SALARY REPORT RUN ON DATE &YYMD"
"  "
PRINT DEPARTMENT CURR_SAL
BY LAST_NAME BY FIRST_NAME
END
```

The resulting output for May 18, 1998 is:

```
SALARY REPORT RUN ON DATE 19990319
```

LAST_NAME	FIRST_NAME	DEPARTMENT	CURR_SAL
-----	-----	-----	-----
BANNING	JOHN	PRODUCTION	\$29,700.00
BLACKWOOD	ROSEMARIE	MIS	\$21,700.00
CROSS	BARBARA	OIS	\$27,062.00
DAVIS	ELIZABETH	MIS	\$.00
GARDNER	DAVID	PRODUCTION	\$.00
GREENSPAN	MARY	MIS	\$9,000.00
IRVING	JOAN	PRODUCTION	\$26,862.00
JONES	DIANE	MIS	\$18,400.00
MCCOY	JOHN	MIS	\$18,400.00
MCKNIGHT	ROGER	PRODUCTION	\$16,100.00
ROMANS	ANTHONY	PRODUCTION	\$21,120.00
SMITH	MARY	MIS	\$13,200.00
	RICHARD	PRODUCTION	\$9,500.00
STEVENS	ALFRED	PRODUCTION	\$11,000.00

Statistical Variables

FOCUS posts many statistics concerning overall operations while a procedure executes in the form of statistical variables. As with system variables, FOCUS can automatically supply values for these variables on request.

Reference

Summary of Statistical Variables

A list of Dialogue Manager statistical variables follows:

Variable	Description
&ACCEPTS	Indicates the number of transactions accepted. This variable applies only to MODIFY requests.
&BASEIO	Indicates the number of input/output operations performed.
&CHNGD	Indicates the number of segments updated. This variable applies only to MODIFY requests.
&DELTDD	Indicates the number of segments deleted. This variable applies only to MODIFY requests.
&DUPLS	Indicates the number of transactions rejected as a result of duplicate values in the data source. This variable applies only to MODIFY requests.
&FOCDISORG	Indicates the percentage of disorganization for a FOCUS file. This variable can be displayed or tested even if the value is less than 30% (the level at which ? FILE displays the amount of disorganization).
&FOCERRNUM	Indicates the last error number, in the format FOCnnnn, displayed after the execution of a procedure. If more than one occurred, &FOCERRNUM will hold the number of the most recent error. If no error occurred, &FOCERRNUM will have a value of 0. This value can be passed to the operating system with the line -QUIT FOCUS &FOCERRNUM. It can also be used to control branching from a procedure to execute an error-handling routine.
&FORMAT	Indicates the number of transactions rejected as a result of a format error. This variable applies only to MODIFY requests.
&INPUT	Indicates the number of segments added to the data source. This variable applies only to MODIFY requests.
&INVALID	Indicates the number of transactions rejected as a result of an invalid condition. This variable applies only to MODIFY requests.
&LINES	Indicates the number of lines printed in last report. This variable applies only to report requests.

Variable	Description
&NOMATCH	Indicates the number of transactions rejected as a result of not matching a value in the data source. This variable applies only to MODIFY requests.
&READS	Indicates the number of records read from a non-FOCUS file.
&RECORDS	Indicates the number of records retrieved in last report. This variable applies only to report requests.
&REJECTS	Indicates the number of transactions rejected for reasons other than the ones specifically tracked by other statistical variables. This variable applies only to MODIFY requests.
&TRANS	Indicates the number of transactions processed. This variable applies only to MODIFY requests.

Example

Using &LINES to Control Execution of a Request

The following example illustrates how to use the statistical variable &LINES to control execution of a request:

```
TABLE FILE SALES
HEADING CENTER
"MONTHLY REPORT FOR &CITY"
SUM UNIT_SOLD AND RETURNS AND COMPUTE
RATIO/D5.2 = 100 * (RETURNS/UNIT_SOLD);
BY CITY
IF CITY EQ &CITY
BY PROD_CODE
IF PROD_CODE IS-FROM &CODE1 TO &CODE2
ON TABLE HOLD
END
-RUN
-IF &LINES EQ 0 GOTO NORECORDS;
MODIFY FILE SALES
.
.
.
DATA ON HOLD
END
-RUN
-NORECORDS
-TYPE No record satisfies this report request
-QUIT
```

In the example, the system calculates the statistical variable &LINES (the number of lines produced by the TABLE request). If the number is 0, there are no lines in the report; -QUIT tells FOCUS to halt processing and the user is returned to the FOCUS prompt. If &LINES is greater than 0, processing continues to the MODIFY request.

Syntax**How to Query the Values of Statistical Variables**

You can query the current value of all statistical variables except &FOCDISORG and &FOCERRNUM by typing the query command

? STAT

from the FOCUS prompt.

Special Variables

FOCUS provides special variables that apply to the cursor, function keys, windows, and other features.

Reference**Summary of Special Variables**

A list of special variables follow:

Variable	Description
&CURSOR	Holds the cursor position.
&CURSORAT	Reads the cursor position.
&ECHO	Controls the display of commands for debugging purposes.
&PFKEY	Holds the PF Key function.
&QUIT	Controls whether the response QUIT, or PF1 in - CRTFORM, to a prompt causes an exit from the procedure.
&STACK	Controls whether the entire procedure, or only the Dialogue Manager commands are executed.
&WINDOWNAME	Holds the name of the last window activated by the most recently executed -WINDOW command (see Chapter 8, <i>Designing Windows With Window Painter</i>).
&WINDOWVALUE	Holds the return value of the last window activated by the most recently executed -WINDOW command (see Chapter 8, <i>Designing Windows With Window Painter</i>).

Using Variables to Alter Commands

A variable can refer to a FOCUS command or to a particular field. In this way, the command structure of a procedure can be determined by the value of the variable.

Example

Using a Field Variable

In the following example, the variable &FIELD determines the field to print in the TABLE request. For example, &FIELD could have the value RETURNS, DAMAGED, or UNIT_SOLD from a file named SALES.

```
TABLE FILE SALES
.
.
.
PRINT &FIELD
BY PROD_CODE
.
.
.
```

Evaluating a Variable Immediately

The .EVAL operator enables you to evaluate a variable's value immediately, making it possible to change a procedure dynamically. It is used for substitution and re-evaluated by Dialogue Manager.

Syntax

How to Evaluate a Variable

.EVAL uses the following syntax

```
[&]&variable.EVAL
```

where:

variable

Is a local or global amper variable.

When the command procedure is executed, the expression is replaced with the value of the specified variable before any other action is performed.

Example**Excluding and Including the .EVAL Operator**

Without the .EVAL operator, an amper variable cannot be used in place of some FOCUS commands, as shown by the following example:

```
-SET &A='-TYPE';
&A HELLO
```

This example's output shows that FOCUS does not recognize the value of &A:

```
UNKNOWN FOCUS COMMAND -TYPE
```

Appending the .EVAL operator to the &A amper variable makes it possible for FOCUS to interpret the variable correctly. For example, adding the .EVAL operator as follows,

```
-SET &A='-TYPE';
&A.EVAL HELLO
```

produces the following output:

```
HELLO
>>
```

Example**Evaluating a Variable Immediately**

The .EVAL operator is particularly useful in modifying code at run time. The following example illustrates how to use the .EVAL operator in a record selection expression. The numbers to the left apply to the notes that follow:

```
1. -SET &R='IF COUNTRY IS ENGLAND';
2. -IF &Y EQ 'YES' THEN GOTO START;
3. -SET &R = '-*';
   -START
4.   TABLE FILE CAR
     PRINT CAR BY COUNTRY
5.   &R.EVAL
   END
```

The procedure executes as follows:

1. The procedure sets the value of &R to 'IF COUNTRY IS ENGLAND'.
2. If the &Y is YES, the procedure branches to the START label, bypassing the second -SET command.
3. If the &Y is NO, the procedure continues to the second -SET command, which sets &R to '-*', which is a comment.
4. The report request is stacked.
5. The procedure evaluates &R's value. If the user wanted a record selection test, &R's value is 'IF COUNTRY IS ENGLAND' and this line is stacked.

If the user did not want a record selection test, &R's value is '-*' and this line is ignored.

Concatenating Variables

You can append a variable to a character string or you can combine two or more variables and/or literals. See the *Creating Reports* manual for full details on concatenation. When using variables, it is important to separate each variable from the concatenation symbol (||) with a space.

Syntax

How to Concatenate Variables

```
-SET &name3 = &name1 || &name2;
```

where:

&name3

Is the name of the concatenated variable.

&name1 || &name2

Are the variables, separated by a space and the concatenation symbol.

Note: The example shown uses strong concatenation, indicated by the || symbol. Strong concatenation removes any trailing blanks from *&name1*. Conversely, weak concatenation, indicated by the symbol |, preserves any trailing blanks in *&name1*.

Creating an Indexed Variable

You can append the value of one variable to the value of another variable, creating an *indexed variable*. This feature applies to both local and global variables.

If the indexed value is numeric, the effect is similar to that of an array in traditional computer programming languages. For example, if the value of index &K varies from 1 to 10, the variable &AMOUNT.&K refers to one of ten variables, from &AMOUNT1 to &AMOUNT10.

A numeric index can be used as a counter; it can be set, incremented, and tested in a procedure.

Syntax

How to Create an Indexed Variable

```
-SET &name.&index[.&index...] = expression;
```

where:

&name

Is a variable.

.&index

Is a numeric or alphanumeric variable whose value is appended to *&name*. The period is required.

When more than one index is used, all index values are concatenated and the string appends to the name of the variable.

For example, &V.&I.&J.&K is equivalent to &V1120 when &I=1, &J=12, and &K=0.

expression

Is a valid expression. For information on the kinds of expressions you can write, see the *Creating Reports* manual.

Example

Using an Indexed Variable in a Loop

An indexed variable can be used in a loop. The following example creates the equivalent of a DO loop used in traditional programming languages:

```
-SET &N = 0;
-LOOP
-SET &N = &N+1;
-IF &N GT 12 GOTO OUT;
-SET &MONTH.&N=&N;
-TYPE &MONTH.&N
-GOTO LOOP
-OUT
```

In this example, &MONTH is the indexed variable and &N is the index. The value of the index is supplied through the command -SET; the first -SET initializes the index to 0, and the second -SET increments the index each time the procedure goes through the loop.

If the value of an index is not defined prior to reference, a blank value is assumed. As a result, the name and value of the indexed variable will not change.

Indexed variables are included in the system limit of 384 variables.

Supplying Values for Variables at Run Time

When you design a Dialogue Manager procedure, you must decide how the variables in the procedure will acquire values. Values for variables can be supplied in two ways:

- When you call a procedure. You can include the variable names and their corresponding values as parameters in an EXEC command that calls one procedure from another.
- Directly in a procedure. The Dialogue Manager commands -DEFAULTS, -SET, and -READ enable you to supply values directly in a procedure.

Example

Supplying Values for Variables

The example in this topic illustrates the use of the commands -DEFAULTS and -SET to supply values for variables. In the example, the user supplies the value of &CODE1, &CODE2, and ®IONMGR as prompted by an HTML form.

The numbers to the left of the example apply to the notes that follow:

```
1.  -DEFAULTS &VERB='SUM'
2.  -SET &CITY=IF &CODE1 GT 'B09' THEN 'STAMFORD' ELSE 'UNIONDALE';
3.  -TYPE REGIONAL MANAGER FOR &CITY
5.      TABLE FILE SALES
          HEADING CENTER
          "MONTHLY REPORT FOR &CITY"
          "PRODUCT CODES FROM &CODE1 TO &CODE2"
          " "
          &VERB UNIT_SOLD AND RETURNS AND COMPUTE
          RATIO/D5.1 = 100 * (RETURNS/UNIT_SOLD);
          BY PROD_CODE
          IF PROD_CODE IS-FROM &CODE1 TO &CODE2
          FOOTING CENTER
4.      "REGION MANAGER: &REGIONMGR"
          "CALCULATED AS OF &DATEMDYY"
          END
6.  -RUN
```

The procedure executes as follows:

1. The -DEFAULTS command sets the value of &VERB to SUM.
2. The -SET command supplies the value for &CITY depending on the value for &CODE1 typed by the user on the form. Because the user typed B10 for &CODE1, the value for &CITY becomes STAMFORD.
3. When the user runs the report, FOCUS writes a message that incorporates the value for &CITY:

```
REGIONAL MANAGER FOR STAMFORD
```

4. The user supplied the value for ®IONMGR on the form. FOCUS supplies the current date at run time.
5. The FOCUS stack contains the following lines:

```
TABLE FILE SALES
HEADING CENTER
"MONTHLY REPORT FOR STAMFORD"
"PRODUCT CODES FROM B10 TO B20"
"  "
SUM UNIT_SOLD AND RETURNS AND COMPUTE
RATIO/D5.1 = 100 * (RETURNS/UNIT_SOLD);
BY PROD_CODE
IF PROD_CODE IS-FROM B10 TO B20
FOOTING CENTER
"REGION MANAGER: SMITH"
"CALCULATED AS OF 03/11/99"
END
```

Reference

General Rules for Supplying Variable Values

The following general rules apply to values for variables:

- The maximum length of a variable value to be displayed on the screen is 80 characters.
- A physical FOCSTACK line with values substituted for variables cannot exceed 80 characters; therefore, you should not use variable values longer than 80 characters.
- If a value contains an embedded space, comma (,) or equal sign (=), you must enclose the variable name in single quotation marks when you use it in an expression. For example, if the value for &CITY is NY, NY, you must refer to the variable as '&CITY' in any expression.
- Once a value is supplied for a local variable, it is used throughout the procedure, unless it is changed by -CRTFORM, -PROMPT, -READ, -SET, or -WINDOW.
- Once a value is supplied for a global variable, it is used throughout the FOCUS session in all procedures, unless it is changed by -CRTFORM, -PROMPT, -READ, -SET, or -WINDOW, or cleared by LET CLEAR.
- Dialogue Manager automatically prompts the terminal if a value has not been supplied for a variable.

Supplying Values Without Prompting

There are several ways to supply values for local and global variables besides prompting methods. These are outlined below:

- Supplying values on the command line: You can supply values when you execute the procedure.
- Supplying values with -DEFAULTS: You supply initial default values in the procedure to ensure that you will not be implicitly prompted for the value.
- Supplying values with -SET: You supply values by setting them in the procedure using the -SET command. The values can be constants or the result of an expression.
- Supplying values with -READ: You can supply values by reading them in from a sequential file.

Supplying Values on the Command Line

When the user knows the values required by a procedure, they can be typed on the command line following the name of the procedure itself. This saves time, since FOCUS now has values to pass to each local or global variable and the user will not be prompted to supply them.

Example

Supplying Values on the Command Line

Consider the following procedure:

```
TABLE FILE SALES
HEADING CENTER
"MONTHLY REPORT FOR &CITY"
SUM UNIT_SOLD AND RETURNS AND COMPUTE
RATIO/D5.2 = 100 * (RETURNS/UNIT_SOLD);
BY PROD_CODE
IF PROD_CODE IS-FROM &CODE1 TO &CODE2
BY CITY
IF CITY EQ &CITY
END
```

In order to execute this procedure and supply values for the variables on the command line, the user would type the following:

```
EX SLRPT CITY = STAMFORD, CODE1=B10, CODE2=B20
```

Syntax

How to Supply Values on the Command Line

Each name-value pair must have the syntactic form

```
name=value
```

and pairs must be separated by commas. It is not necessary to enter the name-value pairs in the order that they are encountered in the procedure.

When the list of values to be supplied exceeds the width of the terminal, insert a comma as the last character on the line and enter the balance of the list on the following line(s), as shown:

```
EX SLRPT AREA=S, CITY = STAMFORD, VERB=COUNT, FIELDS = UNIT_SOLD,  
CODE1=B10, CODE2=B20
```

It is acceptable to supply some but not all values on the command line, in which case, values not supplied will trigger prompts to the terminal.

To supply global amper variable values on the command line, you must supply the double ampersand prefix, as in the following example:

```
EX SLRPT &&GLOBAL=value, CITY = STAMFORD, CODE1=B10, CODE2=B20
```

Example

Using Positional Variables

When the variable is numbered (a positional variable; for example, &1, &2, &3) there is no need to specify the name, in this case a number, on the command line. FOCUS matches the values, one by one to the positional variables as they are encountered in the procedure. Therefore, it is vital to enter the appropriate value for each variable, in the proper order.

Consider the following example:

```
TABLE FILE SALES  
HEADING CENTER  
"MONTHLY REPORT FOR &1"  
SUM UNIT_SOLD AND RETURNS AND COMPUTE  
RATIO/D5.2 = 100 * (RETURNS/UNIT_SOLD);  
BY PROD_CODE  
IF PROD_CODE IS-FROM &2 TO &3  
BY CITY  
IF CITY EQ &1  
END
```

The command line for entry of positional values should read:

```
EX SLRPT STAMFORD, B10, B20
```

Example **Mixing Named and Positional Variables**

You can mix named and positional variables freely on the command line, providing that names are associated with values for named variables and values are supplied for positional variables in the order that these variables are numbered in the procedure. For example:

```
EX SLRPT CITY = STAMFORD, B10, B20, VERB=COUNT
```

Supplying Values With -DEFAULTS

The Dialogue Manager command `-DEFAULTS` supplies an initial (default) value for a variable that had no value before the command was processed. It ensures that values will be passed to variables whether or not they are provided elsewhere.

Syntax **How to Supply Default Values**

```
-DEFAULTS [&]<name>=value [...]
```

where:

`&name`

Is the name of the variable.

`value`

Is the default value assigned to the variable.

Example **Supplying Default Values**

In the following example, `-DEFAULTS` sets default values for `&CITY` and `®IONMGR`.

```
-DEFAULTS &CITY=STAMFORD, &REGIONMGR=SMITH
TABLE FILE SALES
      .
      .
      .
```

Overriding Default Values

You can override default values by supplying new values on the command line or by an explicit prompt.

Supplying Values With -SET

With -SET, you can assign a value computed in an expression.

Syntax

How to Set a Variable Value

```
-SET &[&]name=expression;
```

where:

&name

Is the name of the variable.

expression;

Is a valid literal, arithmetic, or logical expression. Expressions can occupy several lines, so you should end the command with a semicolon (;).

Example

Setting Variable Values

In the following example, -SET assigns the value 14Z or 14B to the variable &STORECODE, as determined by the logical IF expression. The value of &CODE is supplied by the user.

```
-SET &STORECODE = IF &CODE GT C2 THEN '14Z' ELSE '14B';
TABLE FILE SALES
SUM UNIT_SOLD AND RETURNS
BY PROD_CODE
IF PROD_CODE GE &CODE
BY STORE_CODE
IF STORE_CODE IS &STORECODE
END
```

Example

Setting a Literal Value

Single quotation marks around a literal is optional unless it contains embedded blanks, commas, or equal signs, in which case you must include them as shown:

```
-SET &NAME='JOHN DOE';
```

To assign a literal value that includes a single quotation mark, place two single quotation marks where you want one to appear:

```
-SET &NAME='JOHN O 'HARA';
```

Supplying Values With -READ

You can supply values for variables by reading them from a sequential file.

Syntax

How to Supply Values With -READ

```
-READ ddname[, ] [NOCLOSE] &name[.format.][, ] ...
```

where:

ddname

Is the logical name of the file as defined to FOCUS using FILEDEF. (When using MVS, use ALLOCATE or DYNAM ALLOCATE.) A space after the ddname denotes a fixed format file while a comma denotes a comma-delimited file.

NOCLOSE

Indicates that the file should be kept open even if a -RUN is encountered. The file is closed upon completion of the procedure or when a -CLOSE or subsequent -WRITE command is encountered.

name

Is the variable name. You may specify more than one variable. Using commas to separate variables is optional.

If the list of variables is longer than one line, end the first line with a comma and begin the next line with a dash followed by a blank (-). For example:

Comma-delimited files

```
-READ EXTFILE, &CITY,&CODE1,  
- &CODE2
```

Fixed format files

```
-READ EXTFILE &CITY.A8. &CODE1.A3.,  
- &CODE2.A3.
```

format

Is the format of the variable. Note that format must be delimited by periods. The format is ignored for comma-delimited files.

Note: -SET provides an alternate method for defining the length of a variable using the corresponding number of characters enclosed in single quotation marks ('). For example, the following command defines the length of &CITY as 8:

```
-SET &CITY='          ' ;
```

Example**Reading Data and Testing a System Variable**

The example below reads data from EXTFILE, a fixed format file that contains the following data:

```
STAMFORDB10B20
```

The example tests the system variable &IORETURN. If there is no record to be read, the value of &IORETURN is not equal to zero and the procedure branches to the label after the TABLE request.

```
-READ EXTFILE &CITY.A8. &CODE1.A3. &CODE2.A3.
-IF &IORETURN NE 0 GOTO RESUME;
  TABLE FILE SALES
  SUM UNIT_SOLD
  BY CITY
  IF CITY IS &CITY
  BY PROD_CODE
  IF PROD_CODE IS-FROM &CODE1 TO &CODE2
  END
-RESUME
.
.
.
```

Direct Prompting With -PROMPT

The Dialogue Manager command -PROMPT solicits values before the variables to which they refer are used in the procedure. The user is prompted for a value as soon as -PROMPT is encountered. If a looping condition is present, -PROMPT requests a new value for the variable, even if a value exists already. Thus, each time through the loop, the user is prompted for a new value.

With -PROMPT you can specify format, text, and lists in the same way as all other variables.

Example Prompting for Variable Values

The following is an example of the use of -PROMPT:

```
-PROMPT &CODE1
-PROMPT &CODE2
-SET &CITY = IF &CODE1 GT B09 THEN STAMFORD ELSE UNION;
-TYPE REGIONAL MANAGER FOR &CITY
-PROMPT &REGIONMGR
    TABLE FILE SALES
    HEADING CENTER
    "MONTHLY REPORT FOR &CITY"
    "PRODUCT CODES FROM &CODE1 TO &CODE2"
    SUM UNIT_SOLD AND RETURNS AND COMPUTE
    RATIO/D5.2 = 100 * (RETURNS/UNIT_SOLD);
    BY CITY
    IF CITY EQ &CITY
    BY PROD_CODE
    IF PROD_CODE IS-FROM &CODE1 TO &CODE2
    FOOTING CENTER
    "REGION MANAGER: &REGIONMGR"
    "CALCULATED AS OF &DATE"
END
```

-PROMPT sends the following prompts to the screen. User input is shown in lowercase:

PLEASE SUPPLY VALUES REQUESTED

```
CODE1=
b10
CODE2=
b20
REGIONAL MANAGER FOR STAMFORD
REGIONMGR=
smith
>
```

Note how the sequence of supplied values determines the overall flow of the procedure. The value of &CODE1 determines the value of &CITY that gives meaning to the -TYPE command. -TYPE gives the user the necessary information to make the correct choice when supplying the value for ®IONMGR.

By default, all user input is automatically converted to uppercase.

Full-Screen Data Entry With -CRTFORM

-CRTFORM sets up full-screen menus for entering values. The -CRTFORM command in Dialogue Manager and the CRTFORM command in MODIFY are two versions of FIDEL for use in different contexts. The syntax, functions and features are fully outlined in the *Maintaining Databases* manual.

Selecting Data From Menus and Windows With -WINDOW

You can create a series of menus and windows using Window Painter, and then display those menus and windows on the screen using the -WINDOW command. When displayed, the menus and windows can collect data by prompting a user to select a value, to enter a value, or to press a program function (PF) key.

Implied Prompting

If a value is not supplied by any other means for a variable, FOCUS automatically prompts the user for the value. This is known as an implied prompt. These occur sequentially as each variable is encountered in the procedure.

Example

Automatically Prompting for Variable Values

Consider the following example:

```
TABLE FILE SALES
HEADING CENTER
"MONTHLY REPORT FOR &CITY"
.
.
.
BY PROD_CODE
IF PROD_CODE IS-FROM &CODE1 TO &CODE2
.
.
.
FOOTING CENTER
"REGION MANAGER: &REGIONMGR"
"CALCULATED AS OF &DATE"
END
```


When you execute the procedure, FOCUS prompts for the values for the variables one at a time. The terminal dialogue is as follows. User input is in lowercase:

PLEASE SUPPLY VALUES REQUESTED

```
CODE1=  
b10  
CODE2=  
b20  
REGIONMGR=  
smith  
>
```

At the point when all variables have values, FOCUS processes the report request.

Verifying Input Values

Input values can be verified in the following ways:

- Format conditions can be specified against which the entered values are compared.
- Lists of acceptable values can be specified against which the entered values are compared.
- Text can be supplied that either explains what type of value is needed or lists choices of acceptable values on the screen.

Using Format Specifications

You can specify variables with format conditions against which the entered values are compared. If the entered values do not have the specified format, FOCUS prints error messages and prompts the user again for the value(s).

Alphanumeric formats are described by the letter A followed by the number of characters. The number of characters can be from 1 to 255. Integer formats are described by the letter I followed by the number of digits to be entered. The number can be from 1 to 9 (value must be less than $2^{31}-1$).

The description of the format must be enclosed by periods.

If you test field names against input variable values, we recommend that you specify formats of the input variables. If you do not, and the supplied value exceeds the format specification from the Master File, the procedure is ended and error messages are displayed. To continue, the procedure must be executed again. However, if you do include the format, and the supplied value exceeds the format, Dialogue Manager rejects the value and the user is prompted again.

Note: FOCUS internally stores all Dialogue Manager variables as alphanumeric codes. To perform arithmetic operations, Dialogue Manager converts the variable value to double-precision floating point decimal and then converts the result back to alphanumeric codes, dropping the decimal places. For this reason, do not perform tests that look for the decimal places in the numeric codes.

Example

Using a Format Specification

Consider the following format specification:

```
&STORECODE.A3.
```

No special message is sent to the screen detailing the specified format. However, if, in the above example, the user enters more than three alphanumeric characters, the value is rejected, the error message FOC291 is displayed and the user is prompted again.

Note the following example detailing the dialogue between FOCUS and the user:

```
STORECODE=> cc14
```

```
(FOC291) VALUE IN PROMPT REPLY EXCEEDS 03   CHARS:CC14  
STORECODE=>
```

Using Lists of Value Ranges

Variables can be further customized by providing lists of values describing the acceptable range of prompted responses. If the user does not enter one of the available options, the terminal displays the list and re-prompts the user. This is an excellent way to limit the values supplied and to provide help information to the screen while prompting.

Example

Providing a List of Valid Values

For example:

```
-PROMPT &CITY.(STAMFORD,UNIONDALE,NEWARK).
```

A message is printed if the user does not respond with one of the replies on the list. This is followed by a display of the value list. Finally, another prompt is issued for the needed value. For example:

```
CITY>union  
PLEASE CHOOSE ONE OF THE FOLLOWING:  
  STAMFORD, UNIONDALE, NEWARK  
CITY>
```

Syntax

How to Use a Variable to Provide the Reply List

You can also use a variable to provide the reply list, in conjunction with the -SET command. The syntax is

```
-SET &list='value,...';  
-PROMPT &variable.(&list)[.text.]
```

where:

list

Is the name of the reply list variable. Note that in the -PROMPT command, the value is substituted between the parentheses and delimited by periods. If the prompt text has parentheses, enclose that text in single quotation marks (').

value

Is the desired value. You may list more than one value, separated by commas. Enclose the value(s) in single quotation marks ('). A semicolon is required when using -SET.

variable

Is the name of the variable for which you are prompting the user for values.

Example

Using a Variable to Provide the Reply List

For example:

```
-SET &CITIES='STAMFORD,UNIONDALE,NEWARK';  
-PROMPT &CITY.(&CITIES).'(ENTER CITY)'
```

The resulting screen is exactly the same as when the list itself is provided in the parentheses.

You can also create more complex combinations. For example:

```
-SET &CITIES=IF &CODE1 IS B10 THEN 'STAMFORD, NEWARK'  
-ELSE 'STAMFORD, UNIONDALE, NEWARK';
```

Example

Supplying Text for Variable Prompting

A variable can be further specified with customized text explaining the prompt at the screen.

For example:

```
TABLE FILE SALES
HEADING CENTER
"MONTHLY REPORT FOR &CITY. ENTER CITY. "
.
.
.
BY PROD_CODE
IF PROD_CODE IS-FROM &CODE1.A3.BEGINNING CODE. TO
&CODE2.A3.ENDING CODE.
.
.
.
"REGION MANAGER: &REGIONMGR.REGIONAL SUPERVISOR."
"CALCULATED AS OF &DATEMDYY"
END
```

Notice that text has been specified for &CITY and ®IONMGR without specification of a format.

Based on the example, the terminal displays the following prompts one by one:

```
ENTER CITY> stamford
BEGINNING CODE> b10
ENDING CODE> b20
REGIONAL SUPERVISOR> smith
```

Dialogue Manager Quick Reference

This topic describes all the Dialogue Manager commands in alphabetical order. The following commands are included:

-*	-?	-CLOSE
-CMS	-CMS RUN	-CRTCLEAR
-CRTFORM	-DEFAULTS	-EXIT
-GOTO	-IF	-INCLUDE
-label	-MVS RUN	-PASS
-PROMPT	-QUIT	-READ
-REPEAT	-RUN	-SET
-TSO RUN	-TYPE	-WINDOW
-WRITE	-““	

Command: -*

Function: The command -* signals the beginning of a comment line.

Any number of comment lines can follow one another, but each must begin with -*. A comment line may be placed at the beginning or end of a procedure, or in between commands. However, it cannot be on the same line as a command.

Use comment lines liberally to document a procedure so that its purpose and history are clear to others.

Syntax: -** text*

where:

text

Is a comment. A space is not required between -* and text.

Command: -?

Function: The command -? displays the current value of a local variable.

Syntax: -? &[*string*]

where:

string

Is an optional variable name of up to 12 characters. If this parameter is not specified, the current values of all local, global, and defined system and statistical variables are displayed.

Command:	-CLOSE
Function:	-CLOSE closes an external file opened with the -READ or -WRITE NOCLOSE option. The NOCLOSE option keeps a file open until the -READ or -WRITE operation is complete.
Syntax:	<code>-CLOSE {ddname *}</code> where: <i>ddname</i> Is the ddname of the open file described to FOCUS via an allocation (TSO, MSO) or FILEDEF (CMS) command. * Closes all -READ and -WRITE files that are currently open.
Command:	-CMS
Function:	CMS executes a CMS operating system command from within Dialogue Manager.
Syntax:	<code>-CMS command</code> where: <i>command</i> Is a CMS command.
Command:	-CMS RUN
Function:	In CMS, loads and executes the specified user-written function. SET can also execute user-written programs.
Syntax:	<code>-CMS RUN function</code> where: <i>function</i> Is a FOCUS user-written function.
Command:	-CRTCLEAR
Function:	Clears the current screen display.
Syntax:	<code>-CRTCLEAR</code>

Command:	-CRTFORM
Function:	<p>Creates forms to prompt the user for values for variables.</p> <p>All lines following a -CRTFORM command that begin with a hyphen and enclose text in double quotation marks (“”) are part of a single-screen form. Pressing ENTER passes all input data to associated variables.</p> <p>With -CRTFORM, the first line that does not begin with a -“ signals the end of the form. With -CRTFORM BEGIN, the command -CRTFORM END signals the end of the form.</p> <p>All FIDEL facilities are available to -CRTFORM except HEIGHT, WIDTH, and LINE.</p> <p>CRTFORM in MODIFY functions identically to -CRTFORM in Dialogue Manager.</p> <p>See -PROMPT.</p>
Syntax:	<p><code>-CRTFORM [TYPE <i>n</i>] [BEGIN END [LOWER UPPER]]</code></p> <p>where:</p> <p><code>-CRTFORM</code> Invokes FIDEL and signals the beginning of the screen form.</p> <p><code>TYPE <i>n</i></code> Enables you to define the number of lines (<i>n</i>) to reserve for messages. You can specify a number from 1 to 4. The default is 4.</p> <p><code>BEGIN</code> Supports the use of other Dialogue Manager commands to help build the form.</p> <p><code>END</code> Signals the end of the -CRTFORM. Used with -CRTFORM BEGIN.</p> <p><code>LOWER</code> Reads lowercase data from the screen. Once you specify LOWER, every screen thereafter is a lowercase screen until you specify otherwise.</p> <p><code>UPPER</code> Translates lowercase letters to uppercase. This is the default.</p>

Command:	-DEFAULTS
Function:	<p>Sets initial values for the named variables in the procedure.</p> <p>You can override -DEFAULTS values by supplying values for the variables on the command line, by specifically prompting for values with -PROMPT or -CRTFORM, or by supplying a value with -SET subsequent to -DEFAULTS.</p> <p>-DEFAULTS guarantees that the variables are always given a value and therefore that it will execute correctly.</p> <p>Default values are provided in other FOCUS modules to anticipate user needs and reduce the need for keystrokes in situations where most users desire a predefined outcome. See also -SET.</p>
Syntax:	<p><code>-DEFAULTS &name=value, &name=value...</code></p> <p>where:</p> <p><code>name</code> Is the variable name.</p> <p><code>value</code> Is the variable value.</p>
Command:	-EXIT
Function:	<p>-EXIT forces a procedure to end. All stacked commands are executed and the procedure exits (if the procedure was called by another one, the calling procedure continues processing).</p> <p>Use -EXIT for terminating a procedure after processing a final branch that completes the desired task.</p> <p>The last line of a procedure is an implicit -EXIT. In other words, the procedure ends after the last line is read.</p>
Syntax:	<code>-EXIT</code>

Command:	-GOTO
Function:	<p>-GOTO forces an unconditional branch to the specified label.</p> <p>If Dialogue Manager finds the label, processing continues with the line following it.</p> <p>If Dialogue Manager does not find the label, processing ends and an error message is displayed.</p>
Syntax:	<p><code>-GOTO label</code></p> <p><code>.</code></p> <p><code>.</code></p> <p><code>.</code></p> <p><code>-label [TYPE text]</code></p> <p>where:</p> <p><code>label</code></p> <p>Is a user-defined name of up to 12 characters that specifies the target of the -GOTO action.</p> <p>Do not use embedded blanks or the name of any other Dialogue Manager command except -QUIT or -EXIT. Do not use words that can be confused with functions, arithmetic and logical operations, and so on.</p> <p><code>TYPE text</code></p> <p>Optionally sends a message to the client application.</p>
Command:	-HTMLFORM
Function:	For use with the Web Interface to FOCUS.
Syntax:	<code>-HTMLFORM</code>

Command: -IF

Function: -IF routes execution of a procedure based on the evaluation of the specified expression.

An -IF without an explicitly specified ELSE whose expression is false continues processing with the line immediately following it.

Syntax: `-IF expression [THEN] GOTO label1; [ELSE GOTO label2;]
[ELSE IF...;]`

where:

label

Is a user-defined name of up to 12 characters that specifies the target of the GOTO action.

Do not use embedded blanks or the name of any other Dialogue Manager command except -QUIT or -EXIT. Do not use words that can be confused with functions, arithmetic or logical operations, and so on.

expression

Is a valid expression. Literals need not be enclosed in single quotation marks unless they contain embedded blanks or commas.

THEN

Is an optional keyword that increases readability of the command.

ELSE GOTO

Optionally passes control to *label2* when the -IF test fails.

ELSE IF

Optionally specifies a compound -IF test.

The semicolon (;) is required at the end of the command.

Continuation lines must begin with a hyphen (-).

Command:	-INCLUDE
Function:	<p>Specifies another procedure to be attached and executed at run time, as if it were part of the calling procedure. The specified procedure may comprise either a fully developed or partial procedure. Note that a partial procedure does not execute if called outside of the procedure containing -INCLUDE.</p> <p>When using -INCLUDE, you may not branch to a label outside of the specified procedure.</p> <p>A procedure may contain more than one -INCLUDE. Any number of -INCLUDEs may be nested, but recursive -INCLUDEs are not allowed.</p> <p>You may use any valid command in a -INCLUDE.</p> <p>EXEC may also be used to execute a procedure inside another procedure.</p>
Syntax:	<pre>-INCLUDE filename [filetype [filemode]]</pre> <p>where:</p> <p><i>filename</i> Is the procedure to be incorporated in the calling procedure.</p> <p><i>filetype</i> Is the procedure's file type. If none is included, a file type of FOCEXEC is assumed.</p> <p><i>filemode</i> Is the procedure's file mode. If none is included, a file mode of A is assumed.</p>

Command:	-label
Function:	A label is the target of a -GOTO or -IF command.
Syntax:	<code>-label [TYPE message]</code> <p>where:</p> <p><code>label</code></p> <p>Is a user-supplied name of up to 12 characters that identifies the target for a branch.</p> <p>Do not use embedded blanks or the name of any other Dialogue Manager command except -QUIT or -EXIT. Do not use words that can be confused with functions, arithmetic or logical operations, and so on.</p> <p><code>TYPE message</code></p> <p>Optionally sends a message to the client application.</p>
Command:	-MVS RUN
Function:	Same as -TSO RUN.
Syntax:	<code>-MVS RUN</code>
Command:	-PASS
Function:	<p>Passwords can be directly issued and controlled by the Dialogue Manager. This is especially useful to specify a particular file or set of files that a given user can read or write. Passwords have detailed sets of functions associated with them through DBA module.</p> <p>The procedure that sets passwords should be encrypted so that it and the passwords that it sets cannot be typed and made known.</p> <p>A variable can be associated with -PASS so that a password value is prompted for and assigned.</p> <p>The PASS command provides the same function at the command level, as does the PASS parameter of the SET command.</p>
Syntax:	<code>-PASS password</code> <p>where:</p> <p><code>password</code></p> <p>Is a password or a variable containing a password.</p>

Command:	-PROMPT
Function:	<p>-PROMPT types a message to the terminal and reads the reply from the user. This reply assigns a value to the variable named.</p> <p>If a format is specified and the supplied value does not conform, FOCUS displays an error message and prompts the user again for the value.</p> <p>If a (list) is specified and the user does not reply with a value on the list, FOCUS reprompts and prints the list of acceptable values.</p> <p>Note: You cannot use format and list together.</p> <p>In MODIFY, PROMPT specifies additional data input needs.</p> <p>In GRAPH, when it is set on, GPROMPT automatically prompts for all parameters needed to execute the graph request. This is quite a different function from -PROMPT in Dialogue Manager.</p> <p>See -CRTFORM.</p>
Syntax:	<pre>-PROMPT &name [[.format .(<list>)] [.text].]</pre> <p>where:</p> <p><i>&name</i> Is a user-defined variable.</p> <p><i>format</i> Optionally specifies alphanumeric or integer data type and length.</p> <p><i>text</i> Optionally specifies prompting text that appears on the screen. Must be delimited by periods.</p> <p><i>list</i> Optionally specifies a range of acceptable responses. Must be enclosed in parentheses.</p>

Command: -QUIT

Function: Forces an immediate exit from the procedure. Lines that have been stacked are not executed. This differs from an -EXIT, which executes all lines that are currently on the stack.

Like -EXIT, -QUIT returns the user to the FOCUS prompt.

-QUIT FOCUS takes the user out of FOCUS altogether and returns the user to the operating system level.

-QUIT can be made the target of a branch, with the same results as those already described.

QUIT can be entered in response to -PROMPT or -CRTFORM to force an exit from the procedure. The QUIT command can, however, be turned off from within Dialogue Manager to prevent the user from exiting FOCUS prompt.

The QUIT command can also be used to exit from MODIFY and TABLE requests as well as Dialogue Manager procedures.

The principle of QUIT remains consistent throughout FOCUS, namely that the exited request or procedure is not executed and the user is returned to the FOCUS prompt.

See also -RUN and -EXIT.

Syntax: -QUIT or -QUIT FOCUS [*n*]

where:

n

Is the operating system return code. It can be a constant or an integer variable up to 4095. If you do not supply a value or if you supply a non-integer value for *n*, the return code is 8 (the default value).

Command:	-READ
Function:	Reads data from non-FOCUS files. -READ can access data in either fixed or free form. See -WRITE.
Syntax:	<pre>-READ ddname[,] [NOCLOSE] &name[.format.][,] ...</pre> <p>where:</p> <p><i>ddname</i></p> <p>Is the logical name of the file as defined to FOCUS using FILEDEF (or, for MVS, ALLOCATE or DYNAM ALLOCATE). A space after the ddname denotes a fixed format file while a comma denotes a comma-delimited file.</p> <p><i>NOCLOSE</i></p> <p>Indicates that the ddname should be kept open even after a -RUN is executed. The ddname is closed upon completion of the procedure or when a -CLOSE or subsequent -WRITE command is encountered.</p> <p><i>name</i></p> <p>Is the variable name. You may specify more than one variable. Using a comma to separate variables is optional.</p> <p>If the list of variables is longer than one line, end the first line with a comma and begin the next line with a dash followed by a blank (-) for comma-delimited files or a dash followed by a comma followed by a blank (-,) for fixed format files. For example:</p> <p>Comma-delimited files</p> <pre>-READ EXTFILE, &CITY,&CODE1, - &CODE2</pre> <p>Fixed format files</p> <pre>-READ EXTFILE &CITY.A8. &CODE1.A3., -, &CODE2.A3</pre> <p><i>format</i></p> <p>Is the format of the variable. Note that format must be delimited by periods. The format is ignored for comma-delimited files.</p>

Command:	-REPEAT
Function:	<p>-REPEAT allows looping in a procedure.</p> <p>The parameters FROM, TO, and STEP can appear in any order.</p> <p>A loop ends when any of the following occurs:</p> <ul style="list-style-type: none">• It is executed in its entirety.• A -QUIT or -EXIT is issued.• A -GOTO is issued to a label outside of the loop. If a -GOTO is later issued to return to the loop, the loop proceeds from the point it left off.
Syntax:	<pre>-REPEAT <i>label</i> <i>n</i> TIMES -REPEAT <i>label</i> WHILE <i>condition</i> -REPEAT <i>label</i> FOR &<i>variable</i> [FROM <i>fromval</i>] [TO <i>toval</i>] [STEP <i>s</i>]</pre> <p>where:</p> <p><i>label</i></p> <p>Identifies the code to be repeated (the loop). A label can include another loop if the label for the second loop has a different name from the first.</p> <p><i>n</i> TIMES</p> <p>Specifies the number of times to execute the loop. The value of <i>n</i> can be a local variable, a global variable, or a constant. If it is a variable, it is evaluated only once, so the only way to end the loop early is with -QUIT or -EXIT (you cannot change the number of times to execute the loop).</p> <p>WHILE <i>condition</i></p> <p>Specifies the condition under which to execute the loop. The condition is any logical expression that can be true or false. The loop is run if the condition is true.</p> <p>FOR &<i>variable</i></p> <p>Is a variable that is tested at the start of each execution of the loop. It is compared with the value of <i>fromval</i> and <i>toval</i> (if supplied). The loop is executed only if &<i>variable</i> is less than or equal to <i>toval</i> (STEP is positive), or greater than or equal to <i>toval</i> (STEP is negative).</p> <p>FROM <i>fromval</i></p> <p>Is a constant that is compared with &<i>variable</i> at the start of each execution of the loop. The default value is 1.</p>

TO toval

Is a value against which *&variable* is tested. The default is 1,000,000.

STEP s

Is a constant used to increment *&variable* at the end of each execution of the loop. It may be positive or negative. The default value is 1.

Command: -RUN

Function: -RUN causes immediate execution of all stacked FOCUS commands.

Following execution, processing of the procedure continues with the line that follows -RUN.

-RUN is commonly used to do the following:

- Generate results from a request that can then be used in testing and branching.
- Close an external file opened with -READ or -WRITE. When a file is closed, the line pointer is placed at the beginning of the file for a -READ. The line pointer for -WRITE is positioned depending on the allocation and definition of the file.

Syntax: -RUN

Command: -SET

Function: -SET assigns a literal value to a variable, or a value that is computed in an arithmetic or logical expression.

Single quotation marks around a literal value are optional unless it contains embedded blanks or commas, in which case you must include them.

Syntax: -SET *&[&]name=expression;*

where:

&name

Is the name of a variable whose value will be set.

expression

Is a valid expression. Expressions can occupy several lines, so end the command with a semicolon (;).

Command:	-TSO RUN
Function:	<p>In TSO, loads and executes the specified user-written function.</p> <p>Note: The prefix -TSO can be used only with RUN.</p> <p>-SET can also execute user-written programs.</p>
Syntax:	<p><code>-TSO RUN <i>function</i></code></p> <p>where:</p> <p><code><i>function</i></code></p> <p>Is the name of a FOCUS user-written function.</p>
Command:	-TYPE
Function:	<p>Transmits informative messages to the user at the terminal. Any number of -TYPE lines may follow one another but each must begin with -TYPE.</p> <p>Substitutable variables may be embedded in text. The values currently assigned to each variable will be displayed in their assigned position in the text.</p> <p>-TYPE1 and TYPE+ are not supported by IBM 3270-type terminals.</p> <p>TYPE is used in a variety of ways in FOCUS to send informative messages to the screen. A TYPE command may appear on the same line as a label in Dialogue Manager. In MODIFY, TYPE is used to print messages at the start and end of processes, at selected positions in MATCH or NOMATCH, NEXT or NONEXT, and to send a message after an INVALID data condition.</p>
Syntax:	<p><code>-TYPE[+] <i>text</i></code> <code>-TYPE[0] <i>text</i></code> <code>-TYPE[1] <i>text</i></code></p> <p>where:</p> <p><code>-TYPE1</code></p> <p>Sends the text after issuing a page eject.</p> <p><code>-TYPE0</code></p> <p>Sends the text after skipping a line.</p> <p><code>-TYPE+</code></p> <p>Sends the text but does not add a line feed.</p> <p><code><i>text</i></code></p> <p>Is a character string that fits on a line.</p>

Command:	-WINDOW
Function:	<p>Executes a window file. When the command is encountered, control is transferred from the procedure to the specified window file. The window specified in the command becomes the first active window. Control remains within the window file until a menu option is chosen, or a window is activated, for which there is no goto value.</p> <p>The window file, and the windows in it, are created using Window Painter.</p>
Syntax:	<pre>-WINDOW windowfile windowname [PFKEY NOPFKEY] [GETHOLD] [BLANK NOBLANK] [CLEAR NOCLEAR]</pre> <p>where:</p> <p><i>windowfile</i> Identifies the file in which the windows are stored. In CMS, this is a file name. The file must have a file type of FMU. In MVS/TSO, this is a member name. The member must belong to a PDS allocated to ddname FMU.</p> <p><i>windowname</i> Identifies which window in the file will be displayed first.</p> <p><i>PFKEY</i> Enables you to test for function key values during window execution.</p> <p><i>NOPFKEY</i> You are unable to test for function key values during window execution.</p> <p><i>GETHOLD</i> Retrieves stored amper variables collected from a Multi-Select window.</p> <p><i>BLANK</i> Clears all previously set amper variable values when -WINDOW is encountered. This is the default setting.</p> <p><i>NOBLANK</i> When -WINDOW is encountered, the values of previously set amper variables are retained.</p>

CLEAR

Clears the screen before displaying the first window. This is the default behavior. When specified in conjunction with the Terminal Operator Environment (TOE), the TOE screen is redisplayed when control is transferred back to the procedure.

NOCLEAR

Displays the specified window directly over the current screen.

Command:

-WRITE

Function:

Writes information to non-FOCUS files.

Note that all files that have been written should be closed upon any exit from the procedure using -QUIT, -EXIT, or -RUN.

In TABLE, WRITE is a synonym for SUM; functionally it is quite different from -WRITE.

See -READ.

Syntax:

`-WRITE ddname [NOCLOSE] text`

where:

ddname

Is the logical name of the file as defined to FOCUS using FILEDEF (or for MVS, ALLOCATE or DYNAM ALLOCATE).

NOCLOSE

Indicates that the file should be kept open even if a -RUN is encountered. The file is closed upon completion of the procedure or when a -CLOSE or subsequent -READ command is encountered.

text

Is any combination of variables and text. To write more than one line, end the first line with a comma (,) and begin the next line with a hyphen followed by a space (-).

Command _ “ “

Function: The _ “ “ syntax is associated with the FIDEL -CRTFORM command. All textual data enclosed by the double quotation marks is printed to the screen. You can use position markers and specify variable fields within double quotation marks.

When -CRTFORM is processed, the screen displays a form and the cursor stops at each amper variable date entry field. If a variable has not been declared prior to the -CRTFORM, FOCUS prompts the user for a value to assign to the variable.

In MODIFY, enclosing data in double quotation marks (“ “) without the leading hyphen is used with CRTFORM, or for headings, footings, subheads, and subfoots within a TABLE request.

See -CRTFORM.

Syntax: _ “ “

where:

“ “

Enclose textual information, fields and spot markers.

System Defaults and Limits

This topic provides you with an easier way of locating default values, operating system and FOCUS limits, summary tables, general rules, and tips for ease-of-use.

Some general rules to follow when you are creating procedures are:

- If a Dialogue Manager command exceeds one line, the following line must begin with a hyphen (-).
- The hyphen (-) must be placed at the first position of the command line.
- The command is usually attached to the hyphen (-), but you may leave space between the hyphen and the Dialogue Manager command.
- At least one space must be inserted between the Dialogue Manager command and other text.
- Procedure files must have the record format (RECFM) F and the logical record length (LRECL) 80.

The following are some general rules that apply in regard to supplying values for variables:

- The maximum length of a variable value is 79 characters.
- A physical FOCSTACK line with all variables expanded to their full values cannot exceed 80 characters. Since most variables are part of a line in a procedure, it is recommended that you use values that are less than 80 characters long.
- If a value contains an embedded comma (,) or embedded equal sign (=) the value must be enclosed between single quotation marks. For example:

```
EX SLRPT AREA=S, CITY='NY, NY'
```
- Once a value is supplied for a local variable, it is used for that variable throughout the procedure, unless it is changed through a -PROMPT, -SET, or -READ.
- Once a value is supplied for a global variable, it is used for that global variable throughout the FOCUS session in all procedures, unless it is changed through a -PROMPT, -SET, or -READ.
- Dialogue Manager automatically sends a prompt to the terminal if a value has not been supplied for a variable. Automatic prompts (implied prompting) are identical in syntax and function to the direct prompts created with -PROMPT.

The following is a list of operating system default values, limits, and format specifications.

- The default value for the operating system return code value is 8.
- The maximum number of amper variables available in a procedure is 512, of which approximately 30 are reserved for use by FOCUS. This includes all local, global, system, statistical, special, and index variables.
- Literals must be surrounded by single quotation marks if they contain embedded blanks or commas. To produce a literal that includes a single quotation mark, place two single quotation marks where you want one to appear.
- Alphanumeric formats are described by the letter A followed by the number of characters. The number of characters can be from 1 to 3968.
- Integer formats are described by the letter I followed by the number of digits to be entered. The number can be from one to nine digits in length, value must be less than $2^{31}-1$.
- A label is a user-defined name of up to 12 characters. You cannot use blanks and should not use the name of any other Dialogue Manager command. The label may precede or follow GOTO in the procedure.
- A date given to the Dialogue Manager cannot be more than 20 characters long, including spaces.
- The level of nested -INCLUDE files is limited only by available memory.
- The default setting for &QUIT is ON.
- When using Window Painter:
 - Screens should not begin in row 0, column 0, or column 1.
 - The maximum screen size is 22 rows by 77 columns.
 - A File Contents window has a limit of 12K worth of data. This is approximately 150 lines.
 - The maximum number of menu items is 41.
 - File Name windows must have a WIDTH of 24 or greater, or meaningless characters will appear.

CHAPTER 4

Defining a Word Substitution

Topics:

- The LET Command
- Variable Substitution
- Null Substitution
- Multiple-line Substitution
- Recursive Substitution
- Using a LET Substitution in a COMPUTE or DEFINE Command
- Checking Current LET Substitutions
- Interactive LET Query: LET ECHO
- Clearing LET Substitutions
- Saving LET Substitutions in a File
- Assigning Phrases to Function Keys

A LET substitution enables you to define a word to represent other words and phrases. By substituting words for phrases, you can reduce the typing necessary to enter requests (especially when entering phrases repeatedly) and make your requests easier to understand.

The LET Command

The LET command enables you to represent a word or phrase with another word. This reduces the amount of typing necessary for issuing requests, and makes your requests easier to understand. A substitution is especially useful when you use the same phrase repeatedly. Note that you cannot use LET substitutions in Dialogue Manager commands, and substitutions cannot be used in a MODIFY request.

The LET command has a short form and a long form. Use the short form for one or two LET definitions that fit on one line. Otherwise, use the long form.

When you define a word with LET and then use that word in a request, the word is translated into the word or phrase it represents. The result is the same as if you entered the original word or phrase directly. You can substitute any phrase that you enter online unless you are entering a MODIFY request.

A LET substitution lasts until it is cleared or until the request terminates. In WebFOCUS windows version, a substitution lasts until it is cleared or until your session ends. To clear active LET substitutions, issue the LET CLEAR command. To use the same substitutions in many requests, place the LET commands in a stored procedure. If you want to save currently active LET substitutions, use the LET SAVE facility. These substitutions can then be executed later with one short command.

Syntax

How to Make a Substitution (Short Form)

```
LET word = phrase [;word = phrase...]
```

where:

word

Is a string of up to 80 characters with no embedded blanks.

phrase

Is a string of up to 256 characters, which can include embedded blanks. The phrase can also include other special characters, but semicolons and pound signs need special consideration. If the word you are defining appears in the phrase you are replacing, you must enclose it in single quotation marks.

More than one substitution can be defined on the same line by placing a semicolon between definitions.

Example

Making a Substitution (Short Form)

The LET command defines the word WORKREPORT as a substitute for the phrase TABLE FILE EMPLOYEE:

```
LET WORKREPORT = TABLE FILE EMPLOYEE
```

Issuing the following

```
WORKREPORT  
PRINT LAST_NAME  
END
```

results in this request:

```
TABLE FILE EMPLOYEE  
PRINT LAST_NAME  
END
```

The next command includes TABLE as both the word you are defining and as part of the phrase it is replacing. It is enclosed in single quotation marks in the phrase:

```
LET TABLE = 'TABLE' FILE EMPLOYEE
```

More than one word is defined in the following command. The definitions are separated by a semicolon:

```
LET WORKREPORT=TABLE FILE EMPLOYEE; PR=PRINT
```

Syntax

How to Make a Substitution (Long Form)

```
LET  
word = phrase  
.  
.  
.  
END
```

where:

word

Is a string of up to 80 characters with no embedded blanks.

phrase

Is a string of up to 256 characters which can include embedded blanks.

END

Is required to terminate the command.

As shown, LET and END must each be on a separate line.

As with the short form, you can define several words on one line by separating the definitions with a semicolon. A semicolon is not required after the last definition on a line.

Example **Making a Single Substitution (Long Form)**

The following example illustrates a single substitution.

```
LET  
RIGHTNAME = 'STEVENS' OR 'SMITH' OR 'JONES' OR 'BANNING' OR 'MCCOY' OR  
'MCKNIGHT'  
END
```

Example **Making Multiple Substitutions (Long Form)**

The following example illustrates substitutions that span more than one line. Notice that there is no semicolon after the definition PR = PRINT:

```
LET  
WORKREPORT = TABLE FILE EMPLOYEE; PR = PRINT  
RIGHTNAME   = 'STEVENS' OR 'SMITH' OR 'JONES'  
END
```

Example **Defining Substitutions for Translation**

Non-English speakers can use LET commands to translate a request into another language. For example, this request

```
TABLE FILE CAR  
SUM AVE.RCOST OVER AVE.DCOST  
BY CAR ACROSS COUNTRY  
END
```

can be translated into French as:

```
CHARGER FICHIER CAR  
SOMMER AVE.RCOST SUR AVE.DCOST  
PAR CAR TRAVERS COUNTRY  
FIN
```

Variable Substitution

Using the LET command, you can define a word that represents a variable phrase. A variable phrase contains placeholder symbols (carets) to indicate missing elements in the phrase. This allows you to give a phrase different meanings in different requests. Placeholders can be parts of words within phrases. They can also be used to represent system commands.

Placeholders can be unnumbered or numbered. If the placeholders are not numbered, then they are filled from left to right: the first word in the request after the LET-defined word fills the first placeholder, the second word fills the second placeholder, and so on to the last placeholder. If they are numbered, the placeholders are filled in numerical order. If you do not supply enough words to fill all the placeholders, the extra placeholders are null.

Example

Making a Variable Substitution

The command

```
LET UNDERSCORE = ON < > UNDER-LINE
```

contains one placeholder. After issuing this command, you can use the word UNDERSCORE in a request:

```
TABLE FILE EMPLOYEE
PRINT CURR_SAL BY EMP_ID BY HIRE_DATE
UNDERSCORE EMP_ID
END
```

The field name following the LET-defined word supplies the missing value to the placeholder. In the example, EMP_ID follows the defined word UNDERSCORE. This field name is inserted in the placeholder and translates UNDERSCORE EMP_ID as:

```
ON EMP_ID UNDER-LINE
```

Example

Making Multiple Variable Substitutions (Unnumbered)

Issuing the LET command

```
LET TESTNAME = WHERE LAST_NAME IS < > OR < > OR < >
```

and then including the following line in a request

```
TESTNAME 'MCKNIGHT' 'STEVENS' 'BLACKWOOD'
```

translates the line as:

```
WHERE LAST_NAME IS 'MCKNIGHT' OR 'STEVENS' OR 'BLACKWOOD'
```

Notice that the variable phrase needs no placeholder at the end, and could also be code as WHERE LAST_NAME IS <> OR <>. Once all the placeholders are filled, the rest of the definition follows. In this example, the words MCKNIGHT and STEVENS would fill the two placeholders. BLACKWOOD would be left over, so it would follow the variable phrase.

If you do not supply enough words to fill in all the placeholders, the extra placeholders are null. For example, issuing this LET command

```
LET TESTNAME = WHERE LAST_NAME IS < > OR < > OR
```

and then entering this command

```
TESTNAME 'MCCOY'
```

translates the statement into:

```
WHERE LAST_NAME IS 'MCCOY' OR OR
```

This statement is illegal and produces an error message.

Example

Making Multiple Variable Substitutions (Numbered)

The following LET command contains numbered placeholders:

```
LET TESTNAME = WHERE LAST_NAME IS <1> OR <2> OR <3>
```

Therefore, the following line

```
TESTNAME 'STEVENS' 'MCKNIGHT' 'BLACKWOOD'
```

is translated as follows:

```
WHERE LAST_NAME IS 'STEVENS' OR 'MCKNIGHT' OR 'BLACKWOOD'
```

If two placeholders have the same number, both placeholders are filled with the same word. For example, if you issue this LET command

```
LET RANGE = SUM MAX.<1> AND MIN.<1>
```

and this line

```
RANGE SALARY
```

the translated statement is:

```
SUM MAX.SALARY AND MIN.SALARY
```

Example

Making a Variable Substitution in a Phrase

Issuing the following LET command

```
LET BIGGEST = MAX.< >
```

and entering the line

```
WRITE BIGGEST SALARY
```

translates the statement as:

```
WRITE MAX.SALARY
```

Example

Defining a System Command

Each of the following LET commands define a system command in MVS:

```
LET ALFOC = TSO ALLOC F(< >) DA(< > .FOCUS) SHR
```

```
LET LISTMEM = TSO LISTDS < > MEMBERS
```

Null Substitution

With a null substitution, you can use more than one word to represent a phrase. By using more than one word in a request instead of a single word, you can make the request more readable.

You can define a null word using LET. A null word is ignored by the application.

Syntax

How to Define a Null Word

```
LET word=;
```

Example

Defining a Null Word

This LET command defines DISPLAY as a null word:

```
LET  
DISPLAY=;  
AVESAL = SUM AVE.SALARY BY DEPARTMENT  
END
```

In the following request, the word DISPLAY is used in the code DISPLAY AVESAL, for readability, to make clear that the request prints the value represented by AVESAL:

```
TABLE FILE EMPLOYEE  
DISPLAY AVESAL  
WHERE DEPARTMENT IS 'PRODUCTION'  
END
```

The word DISPLAY is ignored and the request is translated as:

```
TABLE FILE EMPLOYEE  
SUM AVE.SALARY BY DEPARTMENT  
WHERE DEPARTMENT IS 'PRODUCTION'  
END
```

Multiple-line Substitution

Many commands, such as END, must appear on a separate line in a report request. To include such a command in a LET definition, place a number sign (#) and a space before the command to indicate a new line. This allows you to substitute one word for several lines of code.

Example

Making Multiple-line Substitutions

This LET command uses the number sign and a space to indicate that a new line is required for the END command:

```
LET HOLDREP = ON TABLE HOLD # END
```

The following request

```
TABLE FILE EMPLOYEE  
SUM AVE.GROSS BY EMP_ID BY PAY_DATE  
HOLDREP
```

is translated as:

```
TABLE FILE EMPLOYEE  
SUM AVE.GROSS BY EMP_ID BY PAY_DATE  
ON TABLE HOLD  
END
```

Recursive Substitution

Recursive substitution allows a phrase in one LET definition to contain a word defined in another LET definition. Recursive substitution can also be used to abbreviate long phrases within LET commands.

Example

Making a Recursive Substitution

In the following LET command

```
LET  
TESTNAME=IF LAST_NAME IS RIGHTNAME  
RIGHTNAME = STEVENS OR MCKNIGHT OR MCCOY  
END
```

the word RIGHTNAME in the phrase in the first definition is defined in the second definition. (Note that the two phrases in the LET command could be reversed.) This LET command is equivalent to:

```
LET  
TESTNAME = IF LAST_NAME IS STEVENS OR MCKNIGHT OR MCCOY  
END
```

Example

Abbreviating a Long Phrase

Consider the following LET command, which illustrates recursive substitution:

```
LET
TESTNAME      =  STEVENS OR SMITH OR MCCOY OR CONT1
CONT1         =  BANNING OR IRVING OR ROMANS OR CONT2
CONT2         =  JONES OR BLACKWOOD
END
```

You can use TESTNAME in this request:

```
TABLE FILE EMPLOYEE
PRINT SALARY BY LAST_NAME
IF LAST_NAME IS TESTNAME
END
```

This is the equivalent of:

```
TABLE FILE EMPLOYEE
PRINT SALARY BY LAST_NAME
IF LAST_NAME IS STEVENS OR SMITH OR MCCOY OR
BANNING OR IRVING OR ROMANS
OR JONES OR BLACKWOOD
END
```

Using a LET Substitution in a COMPUTE or DEFINE Command

A semicolon must follow an expression in a COMPUTE or DEFINE command. To use a LET substitution in a DEFINE or COMPUTE, you must include two semicolons in the LET syntax. You cannot create a LET substitution for a phrase that contains a semicolon.

Example

Using a LET Substitution in a COMPUTE or DEFINE Command

The following LET syntax includes two semicolons, since the substitution will be made in a COMPUTE command:

```
LET
SALTEST = LEVEL/A4 = IF SALARY GT 35000 THEN HIGH
ELSE LOW;;
END
```

Issuing the command

```
AND COMPUTE SALTEST
```

translates the line into

```
AND COMPUTE LEVEL/A4 = IF SALARY GT 35000 THEN HIGH
ELSE LOW;
```

with one semicolon after the word LOW, as required by the expression in the COMPUTE.

Checking Current LET Substitutions

The ? LET command displays the currently active LET substitutions.

Syntax

How to Check Current LET Substitutions

```
? LET [word1 word2 ... wordn]
```

where:

```
word1 word 2...wordn
```

Are the LET-defined words you want to check. If you omit these parameters, ? LET displays a two-column list of all active LET substitutions. The left column contains the LET-defined words; the right column contains the phrases the words represent.

Example

Checking Selected LET Substitutions

Issuing

```
? LET CHART TESTNAME RIGHTNAME
```

displays a two-column list of the LET substitutions for CHART, TESTNAME, and RIGHTNAME.

Example

Checking All Current LET Substitutions

Issuing

```
? LET
```

displays a list of all current LET substitutions.

Interactive LET Query: LET ECHO

The LET ECHO facility shows how FOCUS interprets FOCUS statements. This facility is a diagnostic tool you can use when statements containing LET-defined words are not being interpreted the way you expect them to. Enter:

```
LET ECHO
```

This turns on the LET ECHO facility. When you enter a FOCUS statement, LET ECHO displays the statement as interpreted by FOCUS.

Note:

- If you enter a statement containing no LET-defined words, LET ECHO displays the statement as you entered it.
- If you enter a statement containing LET-defined words, LET ECHO displays the statement with the substitutions made.
- If the statement contains variable substitutions, LET ECHO displays the substitutions with the placeholders filled in.
- If the statement contains multiple-line substitutions, LET ECHO displays the statement with the substitutions on multiple lines.

- If the statement contains null substitutions, LET ECHO displays the statement with the LET-defined words deleted.
- If the statement contains recursive substitutions, the substitutions appear as they are finally resolved.
- LET ECHO may be coded at the top of a FOCEXEC. END ECHO gets coded on the last line of the FOCEXEC.

Note: If you enter a statement containing a variable substitution, you must enter as many words after the LET-defined word as there are placeholders in the phrase; otherwise, LET ECHO will wait for additional input.

Syntax

How to Activate the LET ECHO Facility

LET ECHO facility, issue the command:

```
LET ECHO
```

Syntax

How to Deactivate the LET ECHO Facility

To turn off the LET ECHO facility and return to the FOCUS command level, enter:

```
ENDECHO
```

Clearing LET Substitutions

Use the LET CLEAR command to clear LET substitutions.

Syntax

How to Clear LET Substitutions

```
LET CLEAR {*|word1 [word2...wordn]}
```

where:

*

Clears all substitutions.

word1...wordn

Are the LET-defined words that you want to clear.

Example

Clearing LET Substitutions

Issuing the following command

```
LET CLEAR CHART TESTNAME RIGHTNAME
```

clears substitutions for CHART, TESTNAME, and RIGHTNAME. If there are no additional LET substitutions in effect, the following command would have the same effect:

```
LET CLEAR *
```

Saving LET Substitutions in a File

Since LET substitutions only last the duration of a request, saving them is helpful if you need the same substitutions for another request.

To save LET substitutions currently in effect, use the LET SAVE command.

Syntax

How to Save LET Substitutions

```
LET SAVE [filename]
```

where:

filename

Is the eight-character name of the file in which you want to save the substitutions. If you do not supply a file name, the default file name is LETSAVE.

Assigning Phrases to Function Keys

You can assign a phrase to a function key. Then when you have a blank line and press a function key, that phrase appears as if you actually typed it. This process works only in situations where the LET facility is operative.

Syntax

How to Assign a Phrase to a Function Key

```
LET !n = [.]phrase
```

where:

n

Is a function key number from 1 to 24.

.

Suppresses the echo of the phrase when you press the function key.

phrase

Is the phrase that the specified function key represents.

Example

Assigning Phrases to Function Keys

The following assigns values to function keys:

```
LET !4 = EX DAILYRPT
LET !6 = END
LET !20 = IF RECORDLIMIT EQ 10
LET !21 = .EX MYREPORT
```

CHAPTER 5

Enhancing Application Performance

Topics:

- FOCUS Facilities
- Loading a File
- Compiling a MODIFY Request
- Accessing a FOCUS Data Source (MVS Only)
- Enhancing File Management With HiperFOCUS

This topic covers FOCUS facilities that are available to you across command environment boundaries. These facilities are easy to use and, in many cases, step-by-step instructions are provided.

FOCUS Facilities

The FOCUS facilities discussed in this topic are classified as file utilities for FOCUS and external files. They are summarized in the following table:

Command	Description
LOAD	Loads FOCUS procedures and Master Files into memory (see <i>Loading a File</i> on page 5-2).
COMPILE	Translates MODIFY requests into compiled code ready for execution (see <i>Compiling a MODIFY Request</i> on page 5-7).
MINIO	Note: This facility is for MVS only. Improves performance by reducing I/O operations when accessing FOCUS data sources (see <i>Accessing a FOCUS Data Source (MVS Only)</i> on page 5-8).
SET HIPERFOCUS	Improves performance by using hiperspaces.

Loading a File

Use the LOAD command to load the following types of files into memory for use within a FOCUS session:

- Master Files (MASTER).
- Access Files.
- FOCUS procedures (FOCEXEC).
- Compiled MODIFY requests (FOCCOMP).
- MODIFY requests (MODIFY).

Using memory-resident files decreases execution time because the files do not have to be read from disk. Use the UNLOAD command to remove the files from memory.

Syntax

How to Load a File

`LOAD filetype filename1... [filename2...]`

where:

`filetype`

Specifies the type of file to be loaded (MASTER, access file, FOCEXEC, FOCCOMP, or MODIFY).

`filename1...`

Specifies one or more files to be loaded. Separate the file type and file name(s) with a space.

Example **Loading Multiple Files**

The following command loads the four FOCEXECs CARTEST, FOCMAP1, FOCMAP2, and FOCMAP3 into memory:

```
>LOAD FOCEXEC CARTEST FOCMAP1 FOCMAP2 FOCMAP3
```

A subsequent reference to one of these files during the current FOCUS session will use the loaded, rather than the disk, version.

Syntax **How to Unload a File**

```
UNLOAD [*|filetype] [*| filename1... [filename2...] ]
```

where:

filetype

Specifies the type of file to be unloaded (MASTER, access file, FOCEXEC, MODIFY, or FOCCOMP). To unload all files of all types, use an asterisk.

filename1...

Specifies one or more files to be unloaded. Separate the file type and file name(s) with a space. To unload all files of that file type, use an asterisk.

Example **Unloading Multiple Files**

The following command unloads the two memory-resident FOCEXECs CARTEST and FOCMAP3:

```
>UNLOAD FOCEXEC CARTEST FOCMAP3
```

Any subsequent reference to one of these files will use the disk version.

Loading Master Files, FOCUS Procedures, and Access Files

Loading Master Files, Access Files, and FOCEXECs into memory eliminates the I/Os required to read them each time they are referenced. Whenever FOCUS requires a Master File, Access File, or executes a FOCEXEC, it first looks for a memory-resident MASTER, access file, or FOCEXEC file; if FOCUS cannot find the file in memory, it then searches for a disk version in the normal way.

Reference**Considerations for Loading a Master File, FOCUS Procedure, or Access File**

The following are considerations for loading a Master File, FOCUS procedure, and Access File:

- If you load a Master File, Access File, or a FOCEXEC that has already been loaded into memory, the new copy replaces the old copy.
- Do not load a Master File, Access File, or a FOCEXEC that you are developing, because FOCUS will always use the memory-resident copy of the file (until you reload it), rather than the one you are developing. This is because the copy that you are developing on TED or your system editor is the disk copy, not the memory-resident copy.
- A loaded Master File, Access File, or FOCEXEC requires a maximum of 80 bytes of memory for each of its records plus a small amount of control information, rounded up to a multiple of 4200 bytes.
- The following are the file types for the various Access Files:

Access File	File Type
ADABAS	FOCADBS
DATACOM	FOCDTCM
UDB	FOCSQL
IDMS	FOCIDMS
IMS (IMS=NEW only)	ACCESS
MODEL 204	FOCM204
ORACLE	FOCSQL
SQLDS	FOCSQL
S2K	FOCS2K
SUPRA	ACCESS
TERADATA	FOCSQL
TOTAL	FOCTOTAL

Loading a Compiled MODIFY Request

When you load a compiled MODIFY request, FOCUS loads the FOCCOMP file from disk into memory, then reads and parses the Master File and binds the description to the FOCCOMP file. You may then run the request by issuing the RUN command. The RUN command causes FOCUS to search for a memory-resident FOCCOMP file. If FOCUS cannot find the file, it searches for a disk version in the normal way.

Loading FOCCOMP files not only eliminates the I/Os required to read large FOCCOMP files and their associated Master Files, but also causes another, more subtle effect. When you issue the RUN command to execute a FOCCOMP file from disk, virtual storage must be paged in to accommodate it. If the FOCCOMP file is large, it may require many pages (and a large virtual storage area) in a very short time. If you load the FOCCOMP file first, the initial surge of paging occurs only once at LOAD time. After that, each execution of the loaded file requires a lower paging rate.

Syntax

How to Execute a Compiled Request

`RUN request`

where:

`request`

Is the name of the compiled request stored in memory.

Loading a MODIFY Request

The LOAD MODIFY command is similar to the COMPILE command (described in the *Maintaining Databases* manual) except that instead of writing the compiled output to a FOCCOMP file on disk, FOCUS writes the output into memory as a pre-loaded, compiled MODIFY. FOCUS then reads the Master File associated with the MODIFY command from disk and translates it into an internal table that is tightly bound with the compiled MODIFY. Thus the command

`>LOAD MODIFY NEWTAX`

has substantially the same effect as

`>COMPILE NEWTAX`

`>LOAD FOCCOMP NEWTAX`

except that the compiled code is never written to disk.

After you enter a LOAD MODIFY command, the resulting compiled MODIFY is indistinguishable from code loaded with LOAD FOCCOMP. Thus the UNLOAD MODIFY and ? LOAD MODIFY commands produce exactly the same results as the UNLOAD FOCCOMP and ? LOAD FOCCOMP commands. Note that the UNLOAD FOCCOMP and UNLOAD MODIFY commands unload the bound Master File as well.

When you issue the RUN command to invoke a MODIFY procedure, FOCUS looks for a memory-resident compiled procedure (created by a LOAD FOCCOMP or LOAD MODIFY command) of that name. If the procedure cannot be found, FOCUS then searches for a disk version of the FOCCOMP file in the normal way.

The benefits of the LOAD MODIFY command are that disk space is not used to store the FOCCOMP file, disk I/Os are reduced, the FOCEXEC cannot get out of step with the compiled version, and the paging rate is reduced as it is with FOCCOMP files.

Displaying Information About Loaded Files

The ? LOAD command displays the file type, file name, and resident size of currently loaded files.

Syntax

How to Display Information About Loaded Files

? LOAD [*filetype*]

where:

filetype

Specifies the type of file (MASTER, FOCEXEC, access file, FOCCOMP, or MODIFY) on which information will be displayed. To display information on all memory-resident files, omit file type.

Example

Displaying Information About Loaded Files

Issuing the command

? LOAD

produces information similar to the following:

FILES CURRENTLY LOADED

CAR	MASTER	4200	BYTES
EXPERSON	MASTER	4200	BYTES
CARTEST	FOCEXEC	8400	BYTES

Compiling a MODIFY Request

The COMPILE command translates a MODIFY request stored in a FOCEXEC into an executable code module. This module, like an object code module, cannot be edited by a user. However, it loads faster than the original request because the MODIFY commands have already been interpreted by FOCUS (the initialization time of a compiled MODIFY module can be four to ten times faster than the original request). Compiling a request can save a significant amount of time if the request is large and must be executed repeatedly. You compile the request once, and execute the module as many times as you need it.

Enter the COMPILE command at the FOCUS command level (the FOCUS prompt). To module, use the RUN command from the FOCUS command level.

Syntax

How to Compile MODIFY Request

```
COMPILE focexec [AS module]
```

where:

focexec

Is the name of the FOCEXEC where the request is stored.

module

Is the name of the module. The default is the FOCEXEC name. FOCEXEC names and module names are system dependent.

Syntax

How to Execute a Module

```
RUN module
```

where:

module

Is the name of the module.

You will see no difference in execution between the module and the original request, but it will load much faster.

Reference

Considerations for Compiling a MODIFY Request

The following are considerations for compiling a MODIFY request:

- The FOCEXEC procedure to be compiled may only contain one MODIFY request. It may not contain any other FOCUS, Dialogue Manager, or operating system commands.
- Before compiling a request or executing a module, allocate all input and output files such as transaction files and log files. These allocations must be in effect at run time.
- Before compilation, issue any SET, USE, COMBINE, or JOIN commands necessary to run the request.
- If the data source you are modifying is joined to another file (using the JOIN command) during compilation, it must be joined to the file at run time.
- If you are modifying a combined structure (using the COMBINE command), the structure must be combined both at compilation and at run time.
- FOCEXECs prompt for Dialogue Manager variable values at compilation time. These values cannot be changed at run time.
- If you are using FOCUS security to prevent unauthorized users from executing the request, the password you set at compilation time must be the same one set at run time.

Accessing a FOCUS Data Source (MVS Only)

MINIO is a new I/O buffering technique that improves performance by reducing I/O operations when accessing FOCUS data sources under MVS. With MINIO set on, no block is ever read more than once, and therefore the number of reads performed is the same as the number of tracks present. This results in an overall reduction in elapsed times when reading and writing.

With FOCUS data sources that are not disorganized, MINIO can greatly reduce the number of I/O operations for TABLE and MODIFY commands. I/O reductions of up to fifty percent are achievable with MINIO. The actual reduction varies depending on data source structure and average numbers of children segments per parent segment. By reducing I/O operations, elapsed times for TABLE and MODIFY commands also drop.

Syntax

How to Set MINIO

```
SET MINIO = {ON|OFF}
```

where:

ON

Does not read a block more than once; the number of reads performed will be the same as the number of tracks present. This results in an overall reduction in elapsed times when reading and writing. This value is the default.

OFF

Disables MINIO.

Using MINIO

MINIO reduces CPU time slightly while slightly raising memory utilization. MINIO requires one track I/O buffer per referenced segment type. Between 40K and 48K of above-the-line virtual memory is needed per referenced segment.

When MINIO is enabled, FOCUS decides for each command whether or not to employ it, and which data sources to use it with. It is possible in executing a single command referencing several data sources that MINIO might be used for some but not for others. Data sources accessed via indexes, or physically disordered through online updates, are not candidates for MINIO buffering. Physical disorganization, in this case, means that the sequence of selected records jumps all over the data source, as opposed to progressing steadily forward. When disorganization occurs, MINIO abandons its buffering techniques and resorts to the standard I/O methodology.

When reading data sources, MINIO is used with TABLE, TABLEF, GRAPH, MATCH and during the DUMP phase of the REBUILD command, provided the target data source is not accessed via an index or is physically disorganized.

When writing to data sources, MINIO is used with MODIFY but never with MAINTAIN, provided there is no CRTFORM or COMMIT subcommand. CRTFORMs indicate online transaction processing, which requires that completed transactions be written out to the data source. COMMITs are explicit orders to do so. These events are incompatible with MINIO minimization logic and therefore rule out its use.

As with reads, using MINIO with MODIFY also requires that a data source be accessed sequentially. Attempts to access an index, or update physically disorganized data sources both cause MINIO to be disabled. In addition, frequent repositioning to previously accessed records, even within well-organized data sources, will cause MINIO to be disabled.

Determining if a Previous Command Used MINIO

The ? STAT command is used to determine whether the previous data source access command employed MINIO.

Syntax

How to Determine if a Previous Command Used MINIO

To determine if a previous command used MINIO, issue the command:

? STAT

Example

Determining if a Previous Command Used MINIO

Typing ? STAT generates a screen similar to the following:

STATISTICS OF LAST COMMAND					
RECORDS	=	0	SEGS CHNGD	=	0
LINEs	=	0	SEGS DELTD	=	0
BASEIO	=	87	NOMATCH	=	0
TRACKIO	=	16	DUPLICATES	=	0
SORTIO	=	0	FORMAT ERRORS	=	0
SORT PAGES	=	0	INVALID CONDTs	=	0
READS	=	1	OTHER REJECTS	=	0
TRANSACTIONS	=	1500	CACHE READS	=	0
ACCEPTED	=	1500	MERGES	=	0
SEGS INPUT	=	1500	SORT STRINGS	=	0
INTERNAL MATRIX CREATED: YES			AUTOINDEX USED: NO		
SORT USED: FOCUS			AUTOPATH USED: NO		
MINIO USED: YES					

In the preceding example MINIO USED is displayed as YES. It may also display NO or DISABLED.

- YES means that MINIO buffering has taken place reducing the number of tracks read/written to the FOCUS data source.
- NO, means that MINIO buffering has not taken place.
- DISABLED means that MINIO buffering was started but terminated as no performance gains could be made. This does not mean that the command did not complete successfully. It only indicates that MINIO buffering began and ended during the read/write.

Reference

Restrictions for Using MINIO

Note the following restrictions when you are using the MINIO command:

- When MINIO is used with MODIFY, all CHECK subcommands are ignored. If a MODIFY command terminates abnormally, the condition of the data source is unpredictable, and it should be restored from a backup copy and the update repeated. Since MINIO is designed to minimize I/O during large data source loads and updates, it has no checkpoint or restart facility. If this is unacceptable, set MINIO off.
- MINIO is not used to access data sources through FOCUS Database Servers (formerly called sink machines) or HLI programs.
- MINIO requires the presence of the TRACKIO feature. Meaning, TRACKIO must be set to ON which is the default setting. If TRACKIO is set to OFF, then MINIO is deactivated.
- MINIO buffering starts when the FOCUS data source exceeds 64 pages in size. If this size is never reached, MINIO is never activated.
- If the file being modified UPDATES, INCLUDES, or DELETES a field that is indexed, MINIO is disabled. In other words, FIELDTYPE=I or INDEX=I is coded in the Master File for this field.
- CRTFORM and COMMIT commands disable MINIO.
- MAINTAIN procedures will not use MINIO buffering techniques.
- MINIO is not enabled if the data source is physically disorganized by transaction processing.

Enhancing File Management With HiperFOCUS

The HiperFOCUS option is a group of related features that accelerate FOCUS processing by improving file management performance and controlling resources. These features are:

- **HiperFile** creates temporary files in hiperspaces. For details, see *Creating Temporary Files in Hiperspaces With HiperFile on OS/390* on page 5-16 and *Creating a Temporary Sort File in Hiperspace on CMS* on page 5-19.
- **HiperCache**, under OS/390, creates FOCUS data source cache memory in a hiperspace. For details, see *Creating Cache in Hiperspaces on OS/390* on page 5-19.
- **HiperRule** enables you to dynamically control the use of HiperFOCUS. For details, see *Controlling HiperFOCUS Use With the HiperRule Facility* on page 5-21.

HiperFOCUS enhances file management operations by making use of hiperspaces to reduce I/O and provide additional memory. These enhancements accelerate application performance.

- Under OS/390, HiperFOCUS uses hiperspaces to speed up processing of temporary files and cache memory. MVS/ESA® hiperspaces are page-addressable memory that supplement the primary address space.
- Under CMS, HiperFOCUS uses hiperspaces to handle temporary sort files more efficiently. VM/ESA® hiperspaces are page-addressable storage spaces that supplement the virtual machine's main memory.

Exploiting these memory facilities avoids writing to disk, saving significant I/O time. It also makes more virtual memory available, and transfers data to and from central storage faster.

The HiperBudget feature of HiperFOCUS on OS/390 regulates the use of expanded storage on a system-wide basis. This feature requires that the IBI Subsystem be installed. For information on the subsystem, consult your FOCUS installation guide.

System administrators and application developers can take advantage of the HiperFOCUS options and configuration settings. End users need only activate HiperFOCUS to take advantage of its capabilities.

Activating HiperFOCUS

To use any HiperFOCUS feature, HiperFOCUS must be installed and activated. Once it has been installed, you can activate and deactivate it using the SET HIPERFOCUS command.

If HiperFOCUS is not installed, The SET HIPERFOCUS command is disabled. You can determine if HiperFOCUS is installed and active by issuing a query command. You can also query the HiperFOCUS facility from Dialogue Manager using the system variable &HIPERFOCUS. This four-character variable has the value ON if HiperFOCUS is installed and active; otherwise, it has the value OFF.

Syntax

How to Activate HiperFOCUS

```
SET HIPERFOCUS = {ON|OFF}
```

where:

ON

Activates HiperFOCUS. This is the default.

OFF

Deactivates HiperFOCUS.

Syntax

How to Determine Whether HiperFOCUS is Activated

```
? SET HIPERFOCUS
```

If HiperFOCUS is installed, the command will display the message HIPERFOCUS ON or HIPERFOCUS OFF.

Installing and Configuring HiperFOCUS

HiperFOCUS is installed and configured using SET parameters. These commands must be set in the FOCPARM profile. You can install HiperFOCUS by simply adding SET HIPERINSTALL=ON to the FOCPARM entries and accepting the defaults. However, since this would not establish limits on the number of hiperspaces that FOCUS could create, it is strongly recommended that you establish operating limits using the HiperFOCUS SET parameters. For details on these parameters, see *HiperFOCUS Installation and Configuration Parameters* on page 5-14.

When installing on VM, HiperFOCUS also requires an XC mode virtual machine. To establish XC mode, issue the following command and then re-IPL CMS:

```
CP SET MACHINE XC
```

Once HiperFOCUS has been configured, the end user can activate or deactivate it. No other end-user intervention is required. HiperFOCUS is transparent to normal end-user activity. The only visible change is an increase in application speed.

Reference HiperFOCUS Installation and Configuration Parameters

The following parameters can be set only in the FOCPARM ERRORS file. On OS/390, this file is member FOCPARM of the data set allocated to ddname ERRORS. On CMS, it is the file named FOCPARM ERRORS.

Parameter Name	Description	Default Value
HIPERINSTALL	Installs or disables HiperFOCUS.	OFF
HIPERSPACE	Is the number of (4K) pages to aggregate for hiperspace. This is equivalent to the IBI Subsystem TCBLIM parameter. If both are set, the lower is enforced.	524287 (2GB)
HIPERFILE	Is the maximum number of (4K) pages in an individual hiperspace. This is equivalent to the IBI Subsystem FILELIM parameter. If both are set, the lower is enforced.	524287 (2GB)
HIPERCACHE	Determines the default CACHE size in 4K pages when HiperFOCUS is activated.	256 (1M)
HIPEREXTENTS	Determines the permissible number of extents.	127
HIPERLOCKED	Enables or disables processing of user interface commands such as SET HiperFOCUS.	OFF (allows processing)

Installing HiperBudget on OS/390

On OS/390, the HiperBUDGET facility is installed as part of the IBI Subsystem installation. (The IBI Subsystem provides communication among address spaces running Information Builders products on the same OS/390 system.) HiperBUDGET uses the subsystem to regulate and report on the overall use of hiperspace on that system. It accomplishes this by enforcing pre-defined limits on hiperspace consumption set at the system, server, user and file levels. Limits set at lower levels may never exceed those set at higher levels. HiperBUDGET parameters must be set using the MVS console or by running a special IBI Subsystem job during installation. For more information on these parameters, see *HiperBUDGET Installation Parameters* on page 5-15.

Reference

HiperBUDGET Installation Parameters

The following parameters can be set only in the OS/390 console or during IBI Subsystem installation. See your FOCUS installation guide for information.

Parameter Name	Description
MVSLIM	Is the maximum number of 4K hiperspace pages for all Information Builders products on the operating system. The value -1 specifies no hiperspace limit checking.
SERVLIM	Is the maximum number of 4K hiperspace pages allowed for multiple users on a per server basis. The value -1 specifies no limit/server checking.
TCBLIM	Is the maximum number of 4K hiperspace pages/per user. The value -1 specifies no limit/user checking. This is equivalent to the FOCPARM HIPERSPACE parameter. If both are set, the lower is enforced.
FILELIM	Is the maximum number of 4K hiperspace pages per individual file. The value -1 specifies no limit/file checking. (This is equivalent to the FOCPARM HIPERFILE parameter. If both are set, the lower is enforced.)

Syntax

How to Query HiperBUDGET Limits and Usage

The ? HBUDGET query shows the Hiperspace limits specified and actual utilization statistics, including: limits set at the system, server, user and file levels; the number of busy pages; the number of hyperextents allowed; and the ddnames and sizes of files allocated in hiperspace or spilled to disk.

```
>? HBUDGET
Total system      limit is not set
Total server      limit is not set
Total hiperspace  limit is not set
Single file size limit is  524288 pages
Total amount of busy pages is  616 pages
Number of extents is set to    127

DDname :Reserved :Hiperspace : Spilled :Spill DDn
```

Creating Temporary Files in Hiperspaces With HiperFile on OS/390

FOCUS dynamically allocates certain files to simplify your file management. HiperFile allocates each of these files to its own hiperspaces. This eliminates the need to access a disk, which saves much I/O time and makes data transfer to and from central storage faster. Together, these efficiencies reduce elapsed time and make applications run more quickly. If a file is too large to be created in a hiperspace, HiperFile creates it on disk instead.

For a complete list of dynamically allocated files, refer to the default space allocation table in member IBITABLA of the partitioned data set FOCCTL.DATA.

If you wish, you can override default allocation attributes by explicitly allocating files yourself. If you do so, you can choose to create the file in a hiperspace or on disk using the DYNAM ALLOCATE command. If you wish to control the behavior of HiperFOCUS, you must understand the following:

- Under what circumstances a file might exceed the size of a hiperspace. For details, see *Determining If a File Fits in a Hiperspace on OS/390* on page 5-16.
- How to explicitly allocate a temporary file to disk. For details, see *Explicitly Allocating a Temporary File to Disk* on page 5-17.
- How to determine where a file has been allocated. For details, see *Determining Where a File Has Been Allocated* on page 5-18.
- How to copy a file from a hiperspace to a disk. For details, see *Copying a Hiperspace File to Disk on OS/390* on page 5-18.
- How to improve page handling. For details, see *Improving Page Handling* on page 5-18.

Determining If a File Fits in a Hiperspace on OS/390

When FOCUS allocates a temporary file in a hiperspace, it first verifies that the file's primary extent does not exceed your site's hiperspace limit. If the primary extent is too large, the file is automatically allocated to disk instead.

If you wish to determine if the allocation for a temporary file is within your site's hiperspace limit, you can calculate the file's size, in 4096-byte pages, using a formula. A hiperspace can be up to two gigabytes. However, your site may have specified smaller limits in one of the following ways:

- **System exit.** In the MVS IEFUSI site-defined system exit.
- **Installation.** When installing HiperFOCUS.

Syntax

How to Determine If a File Fits in a Hiperspace on OS/390

If you use **tracks** as your allocation unit:

```
pages=(primary_extent * 12) + ((secondary_extent * operations) * 12)
```

If you use **cylinders** as your allocation unit:

```
pages=(primary_extent * 12 * 15) + ((secondary_extent * operations) * 12 * 15)  
pages
```

Are the number of 4096-byte pages, determined by the result of the formula.

primary_extent

Is the initial amount of space to be allocated.

secondary_extent

Is the amount of space to allocate when the previously allocated space is filled.

operations

Is the number of extend operations. The default value is 127, but your site can specify fewer extents when installing HiperFOCUS.

Note: This formula assumes that your storage device has 12 records per track and 15 tracks per cylinder.

Explicitly Allocating a Temporary File to Disk

HiperFile allocates many of your temporary files to hiperspaces by default. However, you can override default allocations and allocate files on disk rather than in hiperspaces.

When you explicitly allocate a file with the parameter DISP=(NEW,DELETE) for a sequential data set, or DISP=(NEW,DELETE,DELETE) for a partitioned data set, HiperFile verifies that the file's primary extent does not exceed your site's hiperspace limit, and then creates the file in a hiperspace.

If you wish to allocate one of these files to disk instead of to a hiperspace, you can do so by including the following DYNAM parameters:

- If you *do not* wish to identify a particular unit, specify UNIT NOHIPER. For example:

```
DYNAM ALLOC FILE TEMPDSN SP 5 5 CYL UNIT NOHIPER
```

- If you *do* wish to identify a particular unit, specify UNIT *unit_type* HIPER OFF. For example:

```
DYNAM ALLOC FILE TEMPDSN SP 5 5 CYL UNIT SYSDA HIPER OFF
```

See the *Overview and Operating Environments* manual for general FOCUS file allocation instructions.

Determining Where a File Has Been Allocated

You can determine where a file is allocated by issuing a query command. The query command returns information including the following:

- **DEVICE**, which has the value **HIPERFILE** or **DISK** depending upon where the file was allocated.
- **DSNAME**, which has the value **FOCUS.HIPERFILE.NOT.OPENED** if the file was allocated in a hiperspace but has not yet been opened; otherwise, its value is the data set name. If the first qualifier of the data set name is **HIPER**, this file was created in a hiperspace.

Syntax

How to Determine Where a File Has Been Allocated

```
? {MVS|TSO} DDNAME ddname
```

where:

`ddname`

Is the ddname for which you want to see the allocation information.

Copying a Hiperspace File to Disk on OS/390

In some situations you may wish to copy a file from a hiperspace to disk. For example, if you create a series of **HOLD** files, you may need to save one of them to a permanent data set. You can copy a file from a hiperspace to disk using the **DYNAM COPY** command, as described in the *Overview and Operating Environments* manual.

Improving Page Handling

FOCUS can write pages to disk after exhausting the expanded storage allocation with the **HiperFOCUS Spill to Disk** feature. This feature, which eliminates error messages associated with inadequate storage, is activated by setting the **TRACKIO** parameter to **ON**. To check on how many pages are in expanded storage and how many were spilled to disk, issue the **? HIBUDGET** query command described in *How to Query HiperBUDGET Limits and Usage* on page 5-15.

Note: To preserve performance gains achieved through **HiperFOCUS**, it is recommended that you allow only 10% of the pages to spill to disk. If you find that over 10% are on disk, ask your system administrator to allocate additional space for expanded storage.

Creating a Temporary Sort File in Hiperspace on CMS

Under CMS, HiperFile enhances file-management performance by creating the temporary FOCUS sort file—FOCSORT FOCTEMP—in a hiperspace instead of on disk. This saves I/O time by avoiding disk access, and also makes data transfer to and from central storage faster. Together, these efficiencies reduce elapsed time and make your applications run faster.

Syntax

How to Query VM Hiperspace Use

Issue the following command at the FOCUS command prompt:

```
CMS CP Q SPACES
```

Reference

How to Control VM Hiperspace Size

Hiperspaces can be up to two gigabytes. However, your site may have specified a smaller limit in the following ways:

- **XCONFIG.** Including the XCONFIG ADDRSPACE statement in the VM directory entry for a user's virtual machine.
- **Installation.** When installing HiperFOCUS.

Creating Cache in Hiperspaces on OS/390

With HiperFOCUS activated, FOCUS under OS/390 creates FOCUS data source cache memory in a hiperspace.

Standard cache memory stores previously-read FOCUS data source pages in virtual memory, buffering the pages between disk and the internal work area named BINS. When a request needs to read a data source page, it first searches for the page in BINS, followed by cache, and finally looks on disk. By avoiding unnecessary disk I/O, cache accelerates your data retrieval. Cache is most effective when you issue several consecutive requests against the same data source, where the later requests access a subset of the fields accessed by the original request.

HiperFOCUS optimizes FOCUS data source buffering by storing cache pages in a hiperspace instead of in the FOCUS address space. This makes additional memory available, enabling you to store more data source pages in cache; it also reduces the amount of disk I/O and speeds the transfer of data from disk to cache. The end result is faster data source access. When cache is stored in a hiperspace, it is referred to as HiperCache.

Controlling HiperCache

With HiperFOCUS activated, the default cache buffer size is 256 pages (that is, one megabyte). This differs from standard cache, where the default is 0 pages (that is, cache is turned off). You can change the default for HiperFOCUS cache when HiperFOCUS is installed. You can also control HiperCache in the following ways:

- **Activation.** You can turn HiperCache on and off using the CACHE parameter, described in *How to Set Cache* on page 5-20.

If you turn cache on (SET CACHE = *n*), but HiperFOCUS is turned off (SET HIPERFOCUS = OFF), the buffer is allocated as standard cache.

- **Status.** You can check how many cache reads were done by issuing the ? STAT command.
- **Size.** You can change the size of the HiperCache buffer using the CACHE parameter, described in *How to Set Cache* on page 5-20.

When resizing cache (either HiperCache or standard cache), you may wish to wait until all applications have completed. Changing cache size while cache is active reallocates the cache buffer, discarding all data currently stored there.

Syntax

How to Set Cache

The syntax for setting HiperCache and standard cache is

```
SET CACHE = {0|n}
```

where:

0

Allocates no space to cache; cache is inactive. This value is the default.

n

Is the number of 4K pages of contiguous storage allocated to cache memory. The minimum is two pages; the maximum is determined by the amount of memory available. The default for HiperCache is 256 pages (that is, one megabyte). When CACHE is set to any positive value, the cache facility is on.

Controlling HiperFOCUS Use With the HiperRule Facility

If you are responsible for managing system resources, you may wish to control which FOCUS users use HiperFOCUS, and the conditions under which they do so. HiperFOCUS enables you to do this with the HiperRule facility.

The HiperRule facility dynamically controls HiperFOCUS use based on user ID, day, time, FOCUS release, and related information. The HiperRule facility controls HiperFOCUS by the following process:

1. **Write a rule.** A system administrator writes a rule specifying the conditions under which FOCUS users can access HiperFOCUS.
2. **Activate the HiperRule facility.** The administrator activates the HiperFOCUS HiperRule facility by saving the rule as a file. Each system is governed by a single rule.
3. **Evaluate the rule.** The HiperRule facility evaluates the rule each time a user begins a FOCUS session to determine if the user is eligible to access HiperFOCUS. If the user is eligible, access is allowed; otherwise, the user receives a warning message (FOC897) and remains in the FOCUS session without HiperFOCUS.

HiperFOCUS rules are powerful and flexible ways of managing access.

Writing a HiperRule

A rule is a sequence of rule statements. A rule statement can be:

- **A condition** for allowing a user to access HiperFOCUS. The condition is represented as an expression that is evaluated as true or false. A condition can be of two types: ACCEPT (sufficient conditions) and REQUIRE (necessary, but not sufficient conditions).
- **A temporary field definition.** Subsequent rule statements can refer to the temporary field.

A rule must have at least one ACCEPT statement, and can contain an unlimited number of ACCEPT, REQUIRE, and temporary field statements.

Syntax

How to Write a HiperFOCUS Rule

A rule statement can be one of the following

```
REQUIRE = expression;
```

or

```
ACCEPT = expression;
```

or

```
tempfield[/format] = expression;
```

where:

ACCEPT

Defines a condition that is sufficient for allowing access to HiperFOCUS. The use of ACCEPT statements is described in *Evaluating a HiperRule* on page 5-23.

REQUIRE

Defines a condition that—if the HiperRule facility evaluates it—is necessary, but not sufficient, for allowing access to HiperFOCUS.

tempfield

Is the name of the temporary field. It must begin with a letter, and can include any combination of letters, digits, and underscores (_). All letters from A through Z are valid; other letters available in some languages, such as Å and Ç, are not. The name can be up to 66 characters long, and cannot be qualified.

format

Is the field's format. All formats except TX (text) are supported. The default format is D12.2.

expression

Is any valid expression, as described in the *Creating Reports* manual.

Rule expressions can also refer to HiperFOCUS environment variables, such as USERID and HOUROFDAY. These environment variables are described in *Evaluating a HiperRule* on page 5-23.

Example

Writing a HiperRule

The following rule permits HiperFOCUS to be used at all times by the site's FOCUS system administrator, and at non-peak times (before 9:00 AM or after 5:00 PM) by everyone else:

```
ACCEPT = USERID IS 'FOCSYS';  
REQUIRE = (HOUROFDAY LE '09') OR (HOUROFDAY GE '17');
```

Evaluating a HiperRule

The HiperRule facility evaluates rule statements sequentially, beginning with the first statement.

- It terminates the evaluation process if it encounters a true ACCEPT statement or a false REQUIRE statement.
- All expressions following a true ACCEPT statement or a false REQUIRE statement are not evaluated.

The HiperRule facility permits a user to access HiperFOCUS if all of a rule's evaluated REQUIRE statements are true and at least one of its ACCEPT statements is true. A statement is true when its expression is true.

In some situations, you may find it helpful to create a rule statement that is always true or always false. Because the value 1 represents true, and the value 0 represents false, the following statement is true under all conditions:

```
ACCEPT = 1;
```

The following statement is false under all conditions:

```
REQUIRE = 0;
```

Reference

HiperFOCUS Environment Variables

HiperFOCUS provides the following environment variables to use in rule statements.

Name	Description	Format	Value
USERNAME	User ID.	A8	A valid user ID.
TODAYSDATE	Current date.	YMD	yyyy/mm/dd (If you enter a two-digit year as the value, the century value 19 is assumed.)
DAYOFWEEK	Current day of the week.	A8	SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY
TIMEOFDAY	Current time.	A8	hh.mm.ss
HOUROFDAY	Current hour.	A2	00 - 23
OPERATINGSYS	Operating system.	A8	CMS, TSO, MSO, CRJE (MVS batch)
TERMINALTYPE	Terminal type.	A8	3270 (full-screen mode), TTY (line mode), UNKNOWN
FOCUSREL	FOCUS release number.	A16	n.n[n] (for example, 7.2)

Example

Evaluation of a HiperRule

Assume that a user with user ID PGM030 tries to execute an application on April 25, 2001, invoking the following application rule:

```
ACCEPT  = USERID IS 'HFUSER1';  
REQUIRE = TODAYSDATE GE '20010425';  
REQUIRE = TODAYSDATE LE '20010525';  
ACCEPT  = USERID IS 'PGM030';  
ACCEPT  = USERID IS 'PGM040';
```

1. The first ACCEPT statement evaluates as false, so evaluation continues.
2. The two REQUIRE statements evaluate as true, so evaluation continues.
3. The next ACCEPT statement evaluates as true, so evaluation halts and the rule is satisfied.

Storing a HiperRule

The way in which you store a rule depends upon your operating system:

- In **MVS**, create the rule as member HIPERULE of the ERRORS partitioned data set.
- In **CMS**, create the rule as file HIPERULE ERRORS. Only one file with this name may exist on the FOCUS production disk; otherwise, the Resource Governor is disabled.

CHAPTER 6

Working With Cross-Century Dates

Topics:

- When Do You Use the Sliding Window Technique?
- The Sliding Window Technique
- Applying the Sliding Window Technique
- Defining a Global Window With SET
- Defining a Dynamic Global Window With SET
- Querying the Current Global Value of DEFCENT and YRTHRESH
- Defining a File-Level or Field-Level Window in a Master File
- Defining a Window for a Virtual Field
- Defining a Window for a Calculated Value
- Additional Support for Cross-Century Dates

Many existing business applications use two digits to designate a year, instead of four digits. When they receive a value for a year, such as 00, they typically interpret it as 1900, assuming that the first two digits are 19, for the twentieth century. These applications require a way to handle dates when the century changes (for example, from the twentieth to the twenty-first), or when they need to perform comparisons or arithmetic on dates that span more than one century.

The cross-century date feature described in this topic enables the correct interpretation of the century if it is not explicitly provided, or is assumed to be the twentieth. The feature is application-based, that is, it involves modifications to procedures or metadata so that dates are accurately interpreted and processed. The feature is called the sliding window technique.

When Do You Use the Sliding Window Technique?

If your application accesses dates that contain an explicit century, the century is accepted as is. Your application can run correctly across centuries, and you do not need to use the sliding window technique.

If your application accesses dates without explicit centuries, they assume the default value 19. Your application will require remediation, such as the sliding window technique, to ensure the correct interpretation of the century if the default is not valid, and to run as expected in the next century.

This topic does not cover remediation options such as date expansion, which requires that data be changed in the data source to accommodate explicit century values. For a list of Information Builders documentation on remediation, see your latest *Publications Catalog*.

This topic covers the use of the sliding window technique in reporting applications. Details on when to use the sliding window technique are provided later in this topic. It also includes reference information on the use of the technique with FOCUS MODIFY requests. For additional information on implementing this technique with Maintain, see your database maintenance documentation.

The Sliding Window Technique

With the sliding window technique, you do not need to change stored data from a 2-digit year format to a 4-digit year format in order to determine the century. Instead, you can continue storing 2-digit years and expand them when your application accesses them.

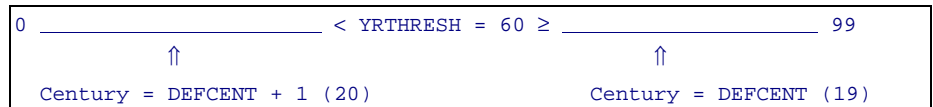
The sliding window technique recognizes that the earliest and latest values for a single date field in most business applications are within 100 years of one another. For example, a human resources application typically contains a field for the birth date of each active employee. The difference in the birth date (or age) of the oldest active employee and the youngest active employee is not likely to be more than 100.

The technique is implemented as follows:

- You define the start of a 100-year sliding window by supplying two values: one for the default century (DEFCENT) and one for the year threshold (YRTHRESH). For example, a value of 19 for the century, combined with a value of 60 for the threshold, creates a window that starts in 1960 and ends in 2059.

- The threshold provides a way to assign a value to the century of a 2-digit year:
 - A year greater than or equal to the threshold assumes the value of the default century (DEFCENT). Using the sample value 19 for the default century and 60 for the threshold, a 2-digit year of 70 is interpreted as 1970 (70 is greater than 60).
 - A year less than the threshold assumes the value of the default century plus 1 (DEFCENT + 1). Using the same sample values (19 and 60), a 2-digit year of 50 is interpreted as 2050 (50 is less than 60), and a 2-digit year of 00 is interpreted as 2000 (00 is also less than 60).

The conversion rule for this example is illustrated as follows:



Any 2-digit year is assumed to fall within the window. You must handle dates that fall outside the defined window by coding.

Each file or each date field used in an application can have its own conversion rule, which provides the flexibility required by most applications.

Defining a Sliding Window

You can define a sliding window in several ways, depending on the specific requirements of your application:

- **Globally.** The SET DEFCENT and SET YRTHRESH commands define a window on a global level.
- **On a file level.** The FDEFCENT and FYRTHRESH attributes in a Master File define a window on a file level, allowing the correct interpretation of date fields from multiple files that span different time periods.
- **On a field level.** The DEFCENT and YRTHRESH attributes in a Master File define a window on a field level, allowing the correct interpretation of date fields, within a single file, that span different time periods.
- **For a virtual field.** The DEFCENT and YRTHRESH parameters on a DEFINE command, in either a request or a Master File, define a window for a virtual field.
- **For a calculated value.** The DEFCENT and YRTHRESH parameters on a COMPUTE command define a window for a calculated value.

If you define more than one window using any of the preceding methods, the precedence is as follows:

1. DEFCENT and YRTHRESH on a DEFINE or COMPUTE command.
2. DEFCENT and YRTHRESH field-level attributes in a Master File.
3. FDEFCENT and FYRTHRESH file-level attributes in a Master File.
4. SET DEFCENT and SET YRTHRESH on a global level; if you do not specify values, the defaults are used (DEFCENT = 19, YRTHRESH = 0).

Creating a Dynamic Window Based on the Current Year

An optional feature of the sliding window technique enables you to create a dynamic window, defining the start of a 100-year span based on the current year. The start year and threshold for the window automatically change at the beginning of each new year.

If an application requires that a window's start year change when a new year begins, use of this feature avoids the necessity of manually re-coding it.

To implement this feature, YRTHRESH or FYRTHRESH is offset from the current year, or given a negative value.

For example, if the current year is 1999 and YRTHRESH is set to -38, a window from 1961 to 2060 is created. The start year 1961 is derived by subtracting 38 (the value of YRTHRESH) from 1999 (the current year). To interpret dates that fall within this window, the threshold 61 is used.

At the beginning of the year 2000, a new window from 1962 to 2061 is automatically created; for dates that fall within this window, the threshold 62 is used. In the year 2001, the window becomes 1963 to 2062, and the threshold is 63, and so on.

With each new year, the start year for the window is incremented by one.

When using this feature, do not code a value for DEFCENT or FDEFCENT, since the feature is designed to automatically calculate the value for the default century. Be aware of the following:

- If you do code a value for DEFCENT on the field level in a Master File, or for FDEFCENT on the file level in a Master File, the feature will not work as intended. The value for the century, which is automatically calculated by YRTHRESH by design, will be reset to the value you code for DEFCENT or FDEFCENT.
- If you code a value for DEFCENT anywhere other than the field level in a Master File (for example, on the global level), and YRTHRESH is negative, the coded value will be ignored. The default century will be automatically calculated as designed.

Applying the Sliding Window Technique

To apply the sliding window technique correctly, you need to understand the difference between a date format (formerly called a smart date) and a legacy date:

- A date format refers to an internally stored integer that represents the number of days between a real date value and a base date (either December 31, 1900, for dates with YMD or YYMD format; or January 1901, for dates with YM, YYM, YQ, or YYQ format). A Master File does not specify a data type or length for a date format; instead, it specifies display options such as D (day), M (month), Y (2-digit year), or YY (4-digit year). For example, MDYY in the USAGE (also known as FORMAT) attribute of a Master File is a date format. A real date value such as March 5, 1999, displays as 03/05/1999, and is internally stored as the offset from December 31, 1900.
- A legacy date refers to an integer, packed decimal, double precision, floating point, or alphanumeric format with date edit options, such as I6YMD, A6MDY, I8YYMD, or A8MDYY. For example, A6MDY is a 6-byte alphanumeric string; the suffix MDY indicates how Information Builders will return the data in the field. The sample value 030599 displays as 03/05/99.

When to Supply Settings for DEFCENT and YRTHRESH

The rest of this topic refers simply to DEFCENT when either DEFCENT or FDEFCENT applies, and to YRTHRESH when either YRTHRESH or FYRTHRESH applies.

Supply settings for DEFCENT and YRTHRESH in the following cases:

- When you issue a DEFINE or COMPUTE command to convert a legacy date without century digits to a date format with century digits (for example, to convert the format I6YMD to YYMD). With DEFINE and COMPUTE, DEFCENT and YRTHRESH do not work directly on legacy dates; for example, you cannot use them to convert the legacy date format I6YMD to the legacy date format I8YYMD.
- When a DEFINE command, COMPUTE command, or Dialogue Manager -SET command calls a function, supplied by Information Builders, that uses legacy dates, and the input date does not contain century digits.

On input, the function will use the window defined for an I6 legacy date field (with edit options). The output format may be I8 (again, with edit options), which includes a 4-digit year.

- When data is entered or changed in a date format field in a FOCUS data source, or an SQL date is entered or changed in a Relational Database Management System (RDBMS), and the input date does not contain century digits.

For example, you can use the sliding window technique in applications that use FIXFORM or CRTFORM with MODIFY.

- When a data source is read, and the ACTUAL attribute in the Master File is non-date specific (for example, A6, I6, or P6), without century digits, and the FORMAT or USAGE attribute specifies a date format. This case does not apply to FOCUS data sources.

Follow these rules when implementing the sliding window technique:

- Specify values for both DEFCEM and YRTHRESH to ensure consistent coding and accurate results, except when YRTHRESH has a negative value. In that case, specify a value for YRTHRESH only; do not code a value for DEFCEM.
- Do not use DEFCEM and YRTHRESH with ON TABLE SET.

Finally, keep in mind that the sliding window technique does not change the way existing data is stored. Rather, it accurately interprets data during application processing.

Reference

Restrictions With MODIFY

The following results occur when you use the sliding window technique with a MODIFY request or FOCCOMP procedure:

- A MODIFY request compiled prior to Version 7.0 Release 6, when run with global SET DEFCEM and SET YRTHRESH settings, or with file-level or field-level settings, yields a FOC1886 error message. You must recompile the MODIFY request.
- A MODIFY request compiled in Version 7.0 Release 6, when run with global SET DEFCEM and SET YRTHRESH settings, or with file-level or field-level settings, yields a FOC1885 warning message.
- A FOCCOMP procedure, compiled with global SET DEFCEM and SET YRTHRESH settings, and run in releases prior to Version 7.0 Release 6, yields a FOC548 invalid version message. You must recompile the MODIFY request.
- A FOCCOMP procedure that contains DEFCEM/YRTHRESH or FDEFCEM/FYRTHRESH attributes in the associated Master File, and run in releases prior to Version 7.0 Release 6, yields a FOC306 description error message.

Date Validation

Date formats are validated on input. For example, 11/99/1999 is rejected as input to a date field formatted as MDYY, because 99 is not a valid day. Information Builders generates an error message.

Legacy dates are not validated. The date 11991999, described with the format A8MDYY, is accepted, even though it, too, contains the invalid day 99.

Defining a Global Window With SET

The SET DEFCEMENT and SET YRTHRESH commands define a window on a global level. The time span created by the SET commands applies to every 2-digit year used by the application unless you specify file-level or field-level windows elsewhere.

For details on specifying parameters that govern the environment, see Chapter 1, *Customizing Your Environment*.

Syntax

How to Define a Global Window With SET

To define a global window, issue two SET commands.

The first command is

```
SET DEFCEMENT = {cc|19}
```

where:

cc

Is the century for the start date of the window. If you do not supply a value, *cc* defaults to 19, for the twentieth century.

The second command is

```
SET YRTHRESH = {[-]yy|0}
```

where:

yy

Is the year threshold for the window. If you do not supply a value, *yy* defaults to zero (0).

If *yy* is a positive number, two-digit years greater than or equal to the threshold default to the value of FDEFCEMENT for the century. Two-digit years less than the threshold assume the value of FDEFCEMENT + 1.

If *yy* is a negative number (-*yy*), the start date of the window is derived by subtracting that number from the current year, and FDEFCEMENT is automatically calculated. The start date is automatically incremented by one at the beginning of each successive year.

Example

Defining a Global Window With SET

In the following request, the SET command defines a global window from 1983 to 2082. As SET syntax allows, the command is entered on one line, with the parameters separated by a comma. You do not need to repeat the keyword SET for YRTHRESH.

The DEFINE command converts the legacy date EFFECT_DATE into the date format NEW_DATE. It creates NEW_DATE as a virtual field, derived from the existing field EFFECT_DATE. The format of EFFECT_DATE is I6YMD, which is a 2-digit year. NEW_DATE is formatted as YYMD, which is a 4-digit year. For details on DEFINE, see the *Creating Reports* manual.

The request is:

```
SET DEFCENT = 19, YRTHRESH = 83

DEFINE FILE EMPLOYEE
NEW_DATE/YYMD = EFFECT_DATE;
END

TABLE FILE EMPLOYEE
PRINT EFFECT_DATE NEW_DATE BY EMP_ID
END
```

In the report, the value of the 2-digit year 82 is less than the threshold 83, so it assumes the value 20 for the century (DEFCENT + 1) and is returned as 2082 in the NEW_DATE column. The other year values (83 and 84) are greater than or equal to the threshold 83, so their century defaults to the value 19 (DEFCENT); they are returned as 1983 and 1984 under NEW_DATE.

The output is:

EMP_ID	EFFECT_DATE	NEW_DATE
-----	-----	-----
071382660		
112847612		
117593129	82/11/01	2082/11/01
119265415		
119329144	83/01/01	1983/01/01
123764317	83/03/01	1983/03/01
126724188		
219984371		
326179357	82/12/01	2082/12/01
451123478	84/09/01	1984/09/01
543729165		
818692173	83/05/01	1983/05/01

In the example, missing date values appear as blanks by default. To retrieve the base date value for the NEW_DATE field instead of blanks, issue the command

```
SET DATEDISPLAY = ON
```

before running the request. The base date value for NEW_DATE, which is formatted as YYMD, is returned as 1900/12/31:

EMP_ID	EFFECT_DATE	NEW_DATE
-----	-----	-----
071382660		1900/12/31
112847612		1900/12/31
117593129	82/11/01	2082/11/01
119265415		1900/12/31
119329144	83/01/01	1983/01/01
123764317	83/03/01	1983/03/01
126724188		1900/12/31
219984371		1900/12/31
326179357	82/12/01	2082/12/01
451123478	84/09/01	1984/09/01
543729165		1900/12/31
818692173	83/05/01	1983/05/01

If NEW_DATE had a YYM format, the base date would appear as 1901/01. If it had a YYQ format, it would appear as 1901 Q1.

If the value of NEW_DATE is 0 and SET DATEDISPLAY = OFF (the default), blanks are displayed. With SET DATEDISPLAY = ON, the base date is displayed instead of blanks. Zero (0) is treated as an offset from the base date, which results in the base date. For details on SET DATEDISPLAY, see Chapter 1, *Customizing Your Environment*.

Defining a Dynamic Global Window With SET

This topic illustrates the creation of a dynamic window using the global command SET YRTHRESH. You can also implement this feature on the file and field level, and on a DEFINE or COMPUTE.

With this option of the sliding window technique, the start year and threshold for the window automatically change at the beginning of each new year. The default century (DEFCENT) is automatically calculated.

You can use SET TESTDATE to alter the system date when testing a dynamic window (that is, when YRTHRESH has a negative value). However, when testing a dynamic window defined in a Master File, you must issue a CHECK FILE command each time you issue a SET TESTDATE command. CHECK FILE reloads the Master File into memory and ensures the correct recalculation of the start date of the dynamic window. For details on SET TESTDATE, see your documentation on the SET command. For details on CHECK FILE, see the *Describing Data* manual.

Example

Defining a Dynamic Global Window With SET

In the following request, the COMPUTE command calls the function AYMD, supplied by Information Builders. AYMD adds one day to the input field, HIRE_DATE; the output field, HIRE_DATE_PLUS_ONE, contains the result. HIRE_DATE is formatted as I6YMD, which is a legacy date with a 2-digit year. HIRE_DATE_PLUS_ONE is formatted as I8YYMD, which is a legacy date with a 4-digit year.

The function uses the YRTHRESH value set at the beginning of the request to create a dynamic window for the input field HIRE_DATE. The start date of the window is incremented by one at the beginning of each new year. Notice that DEFCENT is not coded, since the default century is automatically calculated whenever YRTHRESH has a negative value.

The function inputs a 2-digit year, which is windowed. It then outputs a 4-digit year that includes the century digits.

Sample values are shown in the reports for 1999, 2000, and 2018, which follow the request.

For details on AYMD, see the *Using Functions* manual.

The request is:

```
SET YRTHRESH = -18
```

```
TABLE FILE EMPLOYEE
```

```
PRINT HIRE_DATE AND COMPUTE
```

```
    HIRE_DATE_PLUS_ONE/I8YYMD = AYMD(HIRE_DATE, 1, HIRE_DATE_PLUS_ONE);
```

```
END
```

In 1999, the window spans the years 1981 to 2080. The threshold is 81 (1999 - 18). In the report, the 2-digit year 80 is less than the threshold 81, so it assumes the value 20 for the century (DEFCENT + 1), and is returned as 2080 in the HIRE_DATE_PLUS_ONE column. The other year values (81 and 82) are greater than or equal to the threshold 81, so their century defaults to the value of DEFCENT (19); they are returned as 1981 and 1982.

The output is:

HIRE_DATE	HIRE_DATE_PLUS_ONE
-----	-----
80/06/02	2080/06/03
81/07/01	1981/07/02
82/05/01	1982/05/02
82/01/04	1982/01/05
82/08/01	1982/08/02
82/01/04	1982/01/05
82/07/01	1982/07/02
81/07/01	1981/07/02
82/04/01	1982/04/02
82/02/02	1982/02/03
82/04/01	1982/04/02
81/11/02	1981/11/03

In 2000, the window spans the years 1982 to 2081. The threshold is 82 (2000 - 18). In the report, the 2-digit years 80 and 81 are less than the threshold; for the century, they assume the value 20 (DEFCENT + 1). The 2-digit year 82 is equal to the threshold; for the century, it defaults to the value 19 (DEFCENT).

The output is:

HIRE_DATE	HIRE_DATE_PLUS_ONE
-----	-----
80/06/02	2080/06/03
81/07/01	2081/07/02
82/05/01	1982/05/02
82/01/04	1982/01/05
82/08/01	1982/08/02
82/01/04	1982/01/05
82/07/01	1982/07/02
81/07/01	2081/07/02
82/04/01	1982/04/02
82/02/02	1982/02/03
82/04/01	1982/04/02
81/11/02	2081/11/03

Running the report in 2018 illustrates the automatic recalculation of DEFCENT from 19 to 20. In 2018, the window spans the years 2000 to 2099. The threshold is 0 (2018 - 18). A 2-digit year greater than or equal to 0 defaults to the recalculated value 20 (DEFCENT).

Since all the values for the HIRE_DATE year are greater than 0, their century defaults to 20.

The output is:

HIRE_DATE	HIRE_DATE_PLUS_ONE
-----	-----
80/06/02	2080/06/03
81/07/01	2081/07/02
82/05/01	2082/05/02
82/01/04	2082/01/05
82/08/01	2082/08/02
82/01/04	2082/01/05
82/07/01	2082/07/02
81/07/01	2081/07/02
82/04/01	2082/04/02
82/02/02	2082/02/03
82/04/01	2082/04/02
81/11/02	2081/11/03

Querying the Current Global Value of DEFCENT and YRTHRESH

You can query the current global value of DEFCENT and YRTHRESH.

Syntax

How to Query the Current Global Value of DEFCENT and YRTHRESH

```
? SET DEFCENT
? SET YRTHRESH
```

where:

DEFCENT

Returns the value for the DEFCENT parameter.

YRTHRESH

Returns the value for the YRTHRESH parameter.

Example

Querying the Current Global Value of DEFCENT and YRTHRESH

Enter

```
? SET DEFCENT
? SET YRTHRESH
```

to query the current global value of DEFCENT and YRTHRESH.

The following is a response to the query:

```
DEFCENT      19
YRTHRESH     0
```

Defining a File-Level or Field-Level Window in a Master File

In this implementation of the sliding window technique, you change the metadata used by an application. Two pairs of Master File attributes enable you to define a window on a file or field level:

- The FDEFCENT and FYRTHRESH attributes define a window on a file level. They enable the correct interpretation of legacy date fields from multiple files that span different time periods.
A file-level window takes precedence over a global window for the dates associated with that file.
- The DEFCENT and YRTHRESH attributes define a window on a field level, enabling the correct interpretation of legacy date fields, within a single file, that span different time periods. Each legacy date field in a file can have its own window. For example, in an insurance application, the range of dates for date of birth may be from 1910 to 2009, and the range of dates for expected death may be from 1990 to 2089.
A field-level window takes precedence over a file-level or global window for the dates associated with that field.

For details on Master Files, see the *Describing Data* manual.

Syntax

How to Define a File-Level Window in a Master File

To define a window that applies to all legacy date fields in a file, add the FDEFCENT and FYRTHRESH attributes to the Master File on the file declaration.

The syntax for the first attribute is

```
{FDEFCENT|FDFC} = {cc|19}
```

where:

cc

Is the century for the start date of the window. If you do not supply a value, *cc* defaults to 19, for the twentieth century.

The syntax for the second attribute is

```
{FYRTHRESH|FYRT} = {[ - ]yy|0}
```

where:

yy

Is the year threshold for the window. If you do not supply a value, *yy* defaults to zero (0).

If *yy* is a positive number, two-digit years greater than or equal to the threshold default to the value of DEFCENT for the century. Two-digit years less than the threshold assume the value of DEFCENT + 1.

If *yy* is a negative number (-*yy*), the start date of the window is derived by subtracting that number from the current year, and DEFCENT is automatically calculated. The start date is automatically incremented by one at the beginning of each successive year.

Example**Defining a File-Level Window in a Master File****Tip:**

Use the abbreviated forms of FDEFCENT/FYRTHRESH or DEFCENT/YRTHRESH to reduce keystrokes. The examples in this topic use the abbreviated forms where available (for instance, FDFC instead of FDEFCENT). Maintain supports only the abbreviated forms in certain command syntax (for example, on a COMPUTE or DECLARE command). For details, see the *Maintaining Databases* manual.

In the following example, the FDEFCENT and FYRTHRESH attributes define a window from 1982 to 1981. The window is applied to all legacy date fields in the file, including HIRE_DATE, DAT_INC, and others, if they are converted to a date format.

The Master File is:

```
FILENAME=EMPLOYEE, SUFFIX=FOC, FDFC=19, FYRT=82
SEGNAME=EMPINFO, SEGTYPE=S1
  FIELDNAME=EMP_ID,      ALIAS=EID,      FORMAT=A9,      $
  FIELDNAME=LAST_NAME,  ALIAS=LN,      FORMAT=A15,     $
  FIELDNAME=FIRST_NAME, ALIAS=FN,      FORMAT=A10,     $
  FIELDNAME=HIRE_DATE,  ALIAS=HDT,      FORMAT=I6YMD,   $
.
.
.
  FIELDNAME=DAT_INC,    ALIAS=DI,      FORMAT=I6YMD,   $
.
.
.
```

The DEFINE command in the following request creates two virtual fields named NEW_HIRE_DATE, which is derived from the existing field HIRE_DATE; and NEW_DAT_INC, which is derived from DAT_INC. The format of HIRE_DATE and DAT_INC is I6YMD, which is a legacy date with a 2-digit year. NEW_HIRE_DATE and NEW_DAT_INC are date formats with 4-digit years (YYMD). For details on DEFINE, see the *Creating Reports* manual.

```
DEFINE FILE EMPLOYEE
NEW_HIRE_DATE/YYMD = HIRE_DATE;
NEW_DAT_INC/YYMD = DAT_INC;
END

TABLE FILE EMPLOYEE
PRINT HIRE_DATE NEW_HIRE_DATE DAT_INC NEW_DAT_INC
END
```

The window created in the Master File applies to both legacy date fields. In the report, the year 82 (which is equal to the threshold), for both HIRE_DATE and DAT_INC, defaults to the century value 19 and is returned as 1982 in the NEW_HIRE_DATE and NEW_DAT_INC columns. The year 81, for both HIRE_DATE and DAT_INC, is less than the threshold 82 and assumes the century value 20 (FDEFCENT + 1).

The partial output is:

HIRE_DATE	NEW_HIRE_DATE	DAT_INC	NEW_DAT_INC
80/06/02	2080/06/02	82/01/01	1982/01/01
80/06/02	2080/06/02	81/01/01	2081/01/01
81/07/01	2081/07/01	82/01/01	1982/01/01
82/05/01	1982/05/01	82/06/01	1982/06/01
82/05/01	1982/05/01	82/05/01	1982/05/01
.			
.			
.			

Syntax

How to Define a Field-Level Window in a Master File

To define a window that applies to a specific legacy date field, add the DEFCECENT and YRTHRESH attributes to the Master File on the field declaration.

The syntax for the first attribute is

```
{DEFCECENT|DFC} = {cc|19}
```

where:

cc

Is the century for the start date of the window. If you do not supply a value, *cc* defaults to 19, for the twentieth century.

The syntax for the second attribute is

```
{YRTHRESH|YRT} = {[ -]yy|0}
```

where:

yy

Is the year threshold for the window. If you do not supply a value, *yy* defaults to zero (0).

If *yy* is a positive number, two-digit years greater than or equal to the threshold default to the value of DEFCECENT for the century. Two-digit years less than the threshold assume the value of DEFCECENT + 1.

If *yy* is a negative number (-*yy*), the start date of the window is derived by subtracting that number from the current year, and DEFCECENT is automatically calculated. The start date is automatically incremented by one at the beginning of each successive year.

Example **Defining a Field-Level Window in a Master File**

In this example, the application requires a different window for two legacy date fields in the same file.

The DEFCENT and YRTHRESH attributes in the Master File define a window for HIRE_DATE from 1982 to 2081, and a window for DAT_INC from 1983 to 2082.

The Master File is:

```
FILENAME=EMPLOYEE, SUFFIX=FOC
SEGNAME=EMPINFO, SEGTYPE=S1
  FIELDNAME=EMP_ID,      ALIAS=EID,      FORMAT=A9,      $
  FIELDNAME=LAST_NAME,   ALIAS=LN,      FORMAT=A15,     $
  FIELDNAME=FIRST_NAME,  ALIAS=FN,      FORMAT=A10,     $
  FIELDNAME=HIRE_DATE,   ALIAS=HDT,     FORMAT=I6YMD, DFC=19, YRT=82, $
.
.
.
  FIELDNAME=DAT_INC,     ALIAS=DI,      FORMAT=I6YMD, DFC=19, YRT=83, $
.
.
.
```

The request is the same one used in the previous example (defining a file-level window in a Master File):

```
DEFINE FILE EMPLOYEE
NEW_HIRE_DATE/YYMD = HIRE_DATE;
NEW_DAT_INC/YYMD = DAT_INC;
END

TABLE FILE EMPLOYEE
PRINT HIRE_DATE NEW_HIRE_DATE DAT_INC NEW_DAT_INC
END
```

However, the report illustrates the use of two different windows for the two legacy date fields. For example, the year 82 for HIRE_DATE defaults to the century value 19, since 82 is equal to the threshold for the window for this field. The date returned for NEW_HIRE_DATE is 1982.

The year 82 for DAT_INC assumes the century value 20 (DEFCENT + 1), since 82 is less than the threshold for the window for this field (83). The date returned for NEW_DAT_INC is 2082.

The partial output is:

HIRE_DATE	NEW_HIRE_DATE	DAT_INC	NEW_DAT_INC
80/06/02	2080/06/02	82/01/01	2082/01/01
80/06/02	2080/06/02	81/01/01	2081/01/01
81/07/01	2081/07/01	82/01/01	2082/01/01
82/05/01	1982/05/01	82/06/01	2082/06/01
82/05/01	1982/05/01	82/05/01	2082/05/01
.			
.			

Example**Defining a Field-Level Window in a Master File Used With MODIFY**

This example illustrates the use of field-level DEFCEINT and YRTHRESH attributes to define a window used with MODIFY. To run this example yourself, you need to create a Master File named DATE and a procedure named DATELOAD.

The Master File describes a segment with 12 date fields of different formats. The first field is a date format field. The DEFCEINT and YRTHRESH attributes included on this field create a window from 1990 to 2089. The window is required because the input data for the first date field does not contain century digits, and the default value 19 cannot be assumed.

The Master File looks like this:

```
FILENAME=DATE, SUFFIX=FOC
SEGNAME=ONE, SEGTYPE=S1
  FIELDNAME=D1_YYMD, ALIAS=D1, FORMAT=YYMD, DFC=19, YRT=90, $
  FIELDNAME=D2_I6YMD, ALIAS=D2, FORMAT=I6YMD, $
  FIELDNAME=D3_I8YYMD, ALIAS=D3, FORMAT=I8, $
  FIELDNAME=D4_A6YMD, ALIAS=D4, FORMAT=A6YMD, $
  FIELDNAME=D5_A8YYMD, ALIAS=D5, FORMAT=A8YYMD, $
  FIELDNAME=D6_I4YM, ALIAS=D6, FORMAT=I4YM, $
  FIELDNAME=D7_YQ, ALIAS=D7, FORMAT=YQ, $
  FIELDNAME=D8_YM, ALIAS=D8, FORMAT=YM, $
  FIELDNAME=D9_JUL, ALIAS=D9, FORMAT=JUL, $
  FIELDNAME=D10_Y, ALIAS=D10, FORMAT=Y, $
  FIELDNAME=D11_YY, ALIAS=D11, FORMAT=YY, $
  FIELDNAME=D12_MDYY, ALIAS=D12, FORMAT=MDYY, $
```

The procedure (DATELOAD) creates a FOCUS data source named DATE and loads two records into it. The first field of the first record contains the 2-digit year 92. The first field of the second record contains the 2-digit year 88. For details on commands such as CREATE and MODIFY, and others used in this file, see the *Maintaining Databases* manual.

The procedure looks like this:

```
CREATE FILE DATE
MODIFY FILE DATE
FIXFORM D1/8 D2/6 D3/8 D4/6 D5/8 D6/4 D7/4 D8/4 D9/5 D10/2 D11/4 D12/8
MATCH D1
  ON NOMATCH INCLUDE
  ON MATCH REJECT
DATA
  92022900022920000229000229200002290002000100020006000200002292000
  88022900022920000229000229200002290002000100020006000200002292000
END
```

The following request accesses all the fields in the new data source:

```
TABLE FILE DATE
PRINT *
END
```

In the report, the year 92 for D1_YYMD defaults to the century value 19, since 92 is greater than the threshold for the window for this field (90). It is returned as 1992 in the D1_YYMD column. The year 88 assumes the century value 20 (DEFCENT + 1), because 88 is less than the threshold. It is returned as 2088 in the D1_YYMD column.

The partial output is:

PAGE1

D1_YYMD	D2_I6YMD	D3_I8YYMD	D4_A6YMD	D5_A8YYMD	D6_I4YM	D7_YQ	D8_YM ...
-----	-----	-----	-----	-----	-----	-----	-----
1992/02/29	00/02/29	20000229	00/02/29	2000/02/29	00/02	00 Q1	00/02 ...
2088/02/29	00/02/29	20000229	00/02/29	2000/02/29	00/02	00 Q1	00/02 ...

Example

Defining Both File-Level and Field-Level Windows

The following Master File defines windows at both the file and field level:

```
FILENAME=EMPLOYEE, SUFFIX=FOC, FDFC=19, FYRT=83
SEGNAME=EMPINFO, SEGTYPE=S1
  FIELDNAME=EMP_ID,      ALIAS=EID,      FORMAT=A9,      $
  FIELDNAME=LAST_NAME,   ALIAS=LN,      FORMAT=A15,     $
  FIELDNAME=FIRST_NAME,  ALIAS=FN,      FORMAT=A10,     $
  FIELDNAME=HIRE_DATE,    ALIAS=HDT,     FORMAT=I6YMD, DFC=19, YRT=82, $
.
.
.
  FIELDNAME=EFFECT_DATE,  ALIAS=EDATE,   FORMAT=I6YMD,   $
.
.
.
  FIELDNAME=DAT_INC,      ALIAS=DI,      FORMAT=I6YMD,   $
.
.
.
```

The request is:

```
DEFINE FILE EMPLOYEE
NEW_HIRE_DATE/YYMD = HIRE_DATE;
NEW_EFFECT_DATE/YYMD = EFFECT_DATE;
NEW_DAT_INC/YYMD = DAT_INC;
END

TABLE FILE EMPLOYEE
PRINT HIRE_DATE NEW_HIRE_DATE EFFECT_DATE NEW_EFFECT_DATE DAT_INC
NEW_DAT_INC
END
```

When the field HIRE_DATE is accessed, the time span 1982 to 2081 is applied. For all other legacy date fields in the file, such as EFFECT_DATE and DAT_INC, the time span specified at the file level is applied, that is, 1983 to 2082.

For example, the year 82 for HIRE_DATE is returned as 1982 in the NEW_HIRE_DATE column, since 82 is equal to the threshold of the window for that particular field. The year 82 for EFFECT_DATE and DAT_INC is returned as 2082 in the columns NEW_EFFECT_DATE and NEW_DAT_INC, since 82 is less than the threshold of the file-level window (83).

The partial output is:

HIRE_DATE	NEW_HIRE_DATE	EFFECT_DATE	NEW_EFFECT_DATE	DAT_INC	NEW_DAT_INC
80/06/02	2080/06/02			82/01/01	2082/01/01
80/06/02	2080/06/02			81/01/01	2081/01/01
81/07/01	2081/07/01			82/01/01	2082/01/01
82/05/01	1982/05/01	82/11/01	2082/11/01	82/06/01	2082/06/01
82/05/01	1982/05/01	82/11/01	2082/11/01	82/05/01	2082/05/01

Missing date values for NEW_EFFECT_DATE appear as blanks by default. To retrieve the base date value for NEW_EFFECT_DATE instead of blanks, issue the command

```
SET DATEDISPLAY = ON
```

before running the request. The base date value is returned as 1900/12/31. See *Defining a Global Window With SET* on page 6-7 for sample results.

Defining a Window for a Virtual Field

The DEFCENT and YRTHRESH parameters on a DEFINE command create a window for a virtual field. The window is used to interpret date values for the virtual field when the century is not supplied. You can issue a DEFINE command in either a request or a Master File.

The DEFCENT and YRTHRESH parameters must immediately follow the field format specification; their values are always taken from the left side of the DEFINE syntax (that is, from the left side of the equal sign). If the expression in the DEFINE contains a function call, the function uses the DEFCENT and YRTHRESH values for the input field. The standard order of precedence (field level/file level/global level) applies to the DEFCENT and YRTHRESH values for the input field.

Syntax

How to Define a Window for a Virtual Field in a Request

Use standard DEFINE syntax for a request, as described in your documentation on creating reports. Partial DEFINE syntax is shown here.

On the line that specifies the name of the virtual field, include the DEFCENT and YRTHRESH parameters and values. The parameters must immediately follow the field format information.

```
DEFINE FILE filename  
  fieldname[/format] [{DEFCENT|DFC} {cc|19} {YRTHRESH|YRT} {[-]yy|0}] =  
    expression;
```

```
.  
.   
.   
END
```

where:

filename

Is the name of the file for which you are creating the virtual field.

fieldname

Is the name of the virtual field.

format

Is a date format such as DMY or YYMD.

DEFCENT

Is the parameter for the default century.

cc

Is the century for the start date of the window. If you do not supply a value, *cc* defaults to 19, for the twentieth century.

YRTHRESH

Is the parameter for the year threshold. You must code values for both DEFCENT and YRTHRESH unless YRTHRESH is negative. In that case, only code a value for YRTHRESH.

yy

Is the year threshold for the window. If you do not supply a value, *yy* defaults to zero (0).

If *yy* is a positive number, two-digit years greater than or equal to the threshold default to the value of DEFCENT for the century. Two-digit years less than the threshold assume the value of DEFCENT + 1.

If *yy* is a negative number (-*yy*), the start date of the window is derived by subtracting that number from the current year, and DEFCENT is automatically calculated. The start date is automatically incremented by one at the beginning of each successive year.

expression

Is a valid arithmetic or logical expression or function that determines the value of the virtual field.

END

Is required to terminate the DEFINE command.

Example**Defining a Window for a Virtual Field in a Request**

In the following request, the DEFINE command creates two virtual fields, GLOBAL_HIRE_DATE and WINDOWED_HIRE_DATE. Both virtual fields are derived from the existing field HIRE_DATE. The format of HIRE_DATE is I6YMD, which is a legacy date with a 2-digit year. The virtual fields are date formats with a 4-digit year (YYMD).

The second virtual field, WINDOWED_HIRE_DATE, has the additional parameters DEFCENT and YRTHRESH, which define a window from 1982 to 2081. Notice that both DEFCENT and YRTHRESH are coded, as required.

The request is:

```
DEFINE FILE EMPLOYEE
GLOBAL_HIRE_DATE/YYMD = HIRE_DATE;
WINDOWED_HIRE_DATE/YYMD DFC 19 YRT 82 = HIRE_DATE;
END

TABLE FILE EMPLOYEE
PRINT HIRE_DATE GLOBAL_HIRE_DATE WINDOWED_HIRE_DATE
END
```

Assuming that there are no FDEFCENT and FYRTHRESH file-level settings in the Master File for EMPLOYEE, the global default settings (DEFCENT = 19, YRTHRESH = 0) are used to interpret 2-digit years for HIRE_DATE when deriving the value of GLOBAL_HIRE_DATE. For example, the value of all years for HIRE_DATE (80, 81, and 82) is greater than 0; consequently they default to 19 for the century and are returned as 1980, 1981, and 1982 in the GLOBAL_HIRE_DATE column.

For WINDOWED_HIRE_DATE, the window created specifically for that field (1982 to 2081) is used. The 2-digit years 80 and 81 for HIRE_DATE are less than the threshold for the window (82); consequently, they are returned as 2080 and 2081 in the WINDOWED_HIRE_DATE column.

The output is:

HIRE_DATE	GLOBAL_HIRE_DATE	WINDOWED_HIRE_DATE
-----	-----	-----
80/06/02	1980/06/02	2080/06/02
81/07/01	1981/07/01	2081/07/01
82/05/01	1982/05/01	1982/05/01
82/01/04	1982/01/04	1982/01/04
82/08/01	1982/08/01	1982/08/01
82/01/04	1982/01/04	1982/01/04
82/07/01	1982/07/01	1982/07/01
81/07/01	1981/07/01	2081/07/01
82/04/01	1982/04/01	1982/04/01
82/02/02	1982/02/02	1982/02/02
82/04/01	1982/04/01	1982/04/01
81/11/02	1981/11/02	2081/11/02

Example **Defining a Window for Function Input in a DEFINE Command**

The following sample request illustrates a call to the function AYMD in a DEFINE command. AYMD adds 60 days to the input field, HIRE_DATE; the output field, SIXTY_DAYS, contains the result. HIRE_DATE is formatted as I6YMD, which is a legacy date with a 2-digit year. SIXTY_DAYS is formatted as I8YYMD, which is a legacy date with a 4-digit year.

For details on AYMD, see the *Using Functions* manual.

```
DEFINE FILE EMPLOYEE
SIXTY_DAYS/I8YYMD = AYMD(HIRE_DATE, 60, 'I8YYMD');
END

TABLE FILE EMPLOYEE
PRINT HIRE_DATE SIXTY_DAYS
END
```

The function uses the DEFCENT and YRTHRESH values for the input field HIRE_DATE. In this example, they are set on the field level in the Master File:

```
FILENAME=EMPLOYEE, SUFFIX=FOC
SEGNAME=EMPINFO, SEGTYPE=S1
  FIELDNAME=EMP_ID,      ALIAS=EID,      FORMAT=A9,      $
  FIELDNAME=LAST_NAME,  ALIAS=LN,      FORMAT=A15,     $
  FIELDNAME=FIRST_NAME, ALIAS=FN,      FORMAT=A10,     $
  FIELDNAME=HIRE_DATE,  ALIAS=HDT,      FORMAT=I6YMD, DFC=19, YRT=82, $
.
.
.
```

The function inputs a 2-digit year, which is windowed. It then outputs a 4-digit year that includes the century digits.

The input values 80 and 81 are less than the threshold 82, so they assume the value 20 for the century. The input value 82 is equal to the threshold, so it defaults to 19 for the century.

The output is:

```
PAGE      1

HIRE_DATE  SIXTY_DAYS
-----
80/06/02   2080/08/01
81/07/01   2081/08/30
82/05/01   1982/06/30
82/01/04   1982/03/05
82/08/01   1982/09/30
82/01/04   1982/03/05
82/07/01   1982/08/30
81/07/01   2081/08/30
82/04/01   1982/05/31
82/02/02   1982/04/03
82/04/01   1982/05/31
81/11/02   2082/01/01
```

Syntax

How to Define a Window for a Virtual Field in a Master File

Use standard DEFINE syntax for a Master File, as discussed in your documentation on describing data. Partial DEFINE syntax is shown here.

The parameters DEFCENT and YRTHRESH must immediately follow the field format information.

```
DEFINE fieldname/[format] [{DEFCENT|DFC} {cc|19} {YRTHRESH|YRT} {[-]yy|0}] =  
    expression;$
```

where:

fieldname

Is the name of the virtual field.

format

Is a date format such as DMY or YYMD.

DEFCENT

Is the parameter for the default century.

cc

Is the century for the start date of the window. If you do not supply a value, *cc* defaults to 19, for the twentieth century.

YRTHRESH

Is the parameter for the year threshold. You must code values for both DEFCENT and YRTHRESH unless YRTHRESH is negative. In that case, only code a value for YRTHRESH.

yy

Is the year threshold for the window. If you do not supply a value, *yy* defaults to zero (0).

If *yy* is a positive number, two-digit years greater than or equal to the threshold default to the value of DEFCENT for the century. Two-digit years less than the threshold assume the value of DEFCENT + 1.

If *yy* is a negative number (-*yy*), the start date of the window is derived by subtracting that number from the current year, and DEFCENT is automatically calculated. The start date is automatically incremented by one at the beginning of each successive year.

expression

Is a valid arithmetic or logical expression, function, or function that determines the value of the virtual field.

Example **Defining a Window for a Virtual Field in a Master File**

In the following example, the DEFINE command in a Master File creates a virtual field named NEW_HIRE_DATE. It is derived from the existing field HIRE_DATE. The format of HIRE_DATE is I6YMD, which is a legacy date with a 2-digit year. NEW_HIRE_DATE is a date format with a 4-digit year (YYMD).

The parameters DEFCENT and YRTHRESH on the DEFINE command create a window from 1982 to 2081, which is used to interpret all 2-digit years for the virtual field. Notice that both DEFCENT and YRTHRESH are coded, as required.

The field-level window takes precedence over any global settings in effect. There is no file-level setting in the Master File.

The Master File is:

```
FILENAME=EMPLOYEE, SUFFIX=FOC
SEGNAME=EMPINFO, SEGTYPE=S1
  FIELDNAME=EMP_ID,      ALIAS=EID,      FORMAT=A9,      $
  FIELDNAME=LAST_NAME,  ALIAS=LN,      FORMAT=A15,     $
  FIELDNAME=FIRST_NAME, ALIAS=FN,      FORMAT=A10,     $
  FIELDNAME=HIRE_DATE,  ALIAS=HDT,      FORMAT=I6YMD,   $
.
.
.
DEFINE NEW_HIRE_DATE/YYMD DFC 19 YRT 82 = HIRE_DATE;$
```

The following request generates the values in the sample report:

```
TABLE FILE EMPLOYEE
PRINT HIRE_DATE NEW_HIRE_DATE
END
```

Since the 2-digit years 80 and 81 are less than the threshold 82, their century assumes the value of DEFCENT + 1 (20), and they are returned as 2080 and 2081 in the NEW_HIRE_DATE column. The 2-digit year 82 is equal to the threshold and therefore defaults to the value of DEFCENT (19). It is returned as 1982.

The output is:

HIRE_DATE	NEW_HIRE_DATE
-----	-----
80/06/02	2080/06/02
81/07/01	2081/07/01
82/05/01	1982/05/01
82/01/04	1982/01/04
82/08/01	1982/08/01
82/01/04	1982/01/04
82/07/01	1982/07/01
81/07/01	2081/07/01
82/04/01	1982/04/01
82/02/02	1982/02/02
82/04/01	1982/04/01
81/11/02	2081/11/02

Defining a Window for a Calculated Value

Use the DEFCEM and YRTHRESH parameters on a COMPUTE command in a report request to create a window for a temporary field that is calculated from the result of a PRINT, LIST, SUM, or COUNT command. The window is used to interpret a date value for that field when the century is not supplied.

The DEFCEM and YRTHRESH parameters must immediately follow the field format specification; their values are always taken from the left side of the COMPUTE syntax (that is, from the left side of the equal sign). If the expression in the COMPUTE contains a function call, the function uses the DEFCEM and YRTHRESH values for the input field. The standard order of precedence (field level/file level/global level) applies to the DEFCEM and YRTHRESH values for the input field.

You can also use the parameters on a COMPUTE command in a MODIFY or Maintain procedure, or on a DECLARE command in Maintain. For details on the use of the parameters in Maintain, see the *Maintaining Databases* manual.

Syntax

How to Define a Window for a Calculated Value in a Report

Use standard COMPUTE syntax, as described in your documentation on creating reports. Partial COMPUTE syntax is shown here.

On the line that specifies the name of the calculated value, include the DEFCEM and YRTHRESH parameters and values. The parameters must immediately follow the field format information.

```
TABLE FILE filename
command
[AND] COMPUTE
    fieldname[/format] [{DEFCEM|DFC} {cc|19} {YRTHRESH|YRT} {[-]yy|0}] =
        expression;
.
.
.
END
```

where:

filename

Is the name of the file for which you are creating the calculated value.

command

Is a command such as PRINT, LIST, SUM, or COUNT.

fieldname

Is the name of the calculated value.

format

Is a date format such as DMY or YYMD.

DEFCEM

Is the parameter for the default century.

cc

Is the century for the start date of the window. If you do not supply a value, *cc* defaults to 19, for the twentieth century.

YRTHRESH

Is the parameter for the year threshold. You must code values for both *DEFCENT* and *YRTHRESH* unless *YRTHRESH* is negative. In that case, only code a value for *YRTHRESH*.

yy

Is the year threshold for the window. If you do not supply a value, *yy* defaults to zero (0).

If *yy* is a positive number, two-digit years greater than or equal to the threshold default to the value of *DEFCENT* for the century. Two-digit years less than the threshold assume the value of *DEFCENT* + 1.

If *yy* is a negative number (-*yy*), the start date of the window is derived by subtracting that number from the current year, and *DEFCENT* is automatically calculated. The start date is automatically incremented by one at the beginning of each successive year.

expression

Is a valid arithmetic or logical expression, function, or function that determines the value of the temporary field.

END

Is required to terminate the request.

Syntax

How to Define a Window for a Calculated Value in a MODIFY Request

Use standard MODIFY and COMPUTE syntax, as described in your database maintenance documentation; partial syntax is shown here.

On the line that specifies the name of the calculated value, include the DEFCENT and YRTHRESH parameters and values. The parameters must immediately follow the field format information.

```
MODIFY FILE filename
.
.
.
COMPUTE
  fieldname[/format] [{DEFCENT|DFC} {cc|19} {YRTHRESH|YRT} {[-]yy|0}] =
    expression;
.
.
.
[END]
```

where:

filename

Is the name of the file you are modifying.

fieldname

Is the name of the field being set to the value of *expression*.

format

Is a date format such as MDY or YYMD.

DEFCENT

Is the parameter for the default century.

cc

Is the century for the start date of the window. If you do not supply a value, *cc* defaults to 19, for the twentieth century.

YRTHRESH

Is the parameter for the year threshold. You must code values for both DEFCENT and YRTHRESH unless YRTHRESH is negative. In that case, only code a value for YRTHRESH.

yy

Is the year threshold for the window. If you do not supply a value, *yy* defaults to zero (0).

If *yy* is a positive number, two-digit years greater than or equal to the threshold default to the value of DEFCENT for the century. Two-digit years less than the threshold assume the value of DEFCENT + 1.

If *yy* is a negative number (-*yy*), the start date of the window is derived by subtracting that number from the current year, and DEFCENT is automatically calculated. The start date is automatically incremented by one at the beginning of each successive year.

expression

Is a valid arithmetic or logical expression or function that determines the value of *fieldname*.

END

Terminates the request. Do not add this command if the request contains PROMPT statements.

Example

Defining a Window for a Calculated Value

In the following request, the parameters DEFCENT and YRTHRESH on the COMPUTE command define a window from 1999 to 2098. Notice that both DEFCENT and YRTHRESH are coded, as required. The window is applied to the field created by the COMPUTE command, LATEST_DAT_INC.

DAT_INC is formatted as I6YMD, which is a legacy date with a 2-digit year. LATEST_DAT_INC is a date format with a 4-digit year (YYMD). The prefix MAX retrieves the highest value of DAT_INC.

The request is:

```
TABLE FILE EMPLOYEE
SUM SALARY AND COMPUTE
    LATEST_DAT_INC/YYMD DFC 19 YRT 99 = MAX.DAT_INC;
END
```

The highest value of DAT_INC is 82/08/01. Since the year 82 is less than the threshold 99, it assumes the value 20 for the century (DEFCENT + 1).

The output is:

SALARY	LATEST_DAT_INC
-----	-----
\$332,929.00	2082/08/01

Example**Defining a Window for Function Input in a COMPUTE Command**

The following sample request illustrates a call to the function JULDAT in a COMPUTE command. JULDAT converts dates from Gregorian format (year/month/day) to Julian format (year/day). For century display, dates in Julian format are 7-digit numbers. The first 4 digits are the century. The last three digits represent the number of days, counting from January 1.

For details on JULDAT, see your documentation on creating reports.

In the request, the input field is HIRE_DATE. The function converts it to Julian format and returns it as JULIAN_DATE. HIRE_DATE is formatted as I6YMD, which is a legacy date with a 2-digit year. JULIAN_DATE is formatted as I7, which is a legacy date with a 4-digit year.

```
TABLE FILE EMPLOYEE
PRINT DEPARTMENT HIRE_DATE
AND COMPUTE
      JULIAN_DATE/I7 = JULDAT(HIRE_DATE, JULIAN_DATE);
BY LAST_NAME BY FIRST_NAME
END
```

The function uses the FDEFCENT and FYRTHRESH values for the input field HIRE_DATE. In this example, they are set on the file level in the Master File:

```
FILENAME=EMPLOYEE, SUFFIX=FOC, FDFC=19, FYRT=82
SEGNAME=EMPINFO, SEGTYPE=S1
  FIELDNAME=EMP_ID,      ALIAS=EID,      FORMAT=A9,      $
  FIELDNAME=LAST_NAME,   ALIAS=LN,      FORMAT=A15,     $
  FIELDNAME=FIRST_NAME,  ALIAS=FN,      FORMAT=A10,     $
  FIELDNAME=HIRE_DATE,   ALIAS=HDT,     FORMAT=I6YMD,   $
.
.
.
```

The function inputs a 2-digit year, which is windowed. It then outputs a 4-digit year that includes the century digits.

The input values 80 and 81 are less than the threshold 82, so they assume the value 20 for the century. The input value 82 is equal to the threshold, so it defaults to 19 for the century.

The output follows. By default, the second occurrence of the last name SMITH displays as blanks.

LAST_NAME	FIRST_NAME	DEPARTMENT	HIRE_DATE	JULIAN_DATE
-----	-----	-----	-----	-----
BANNING	JOHN	PRODUCTION	82/08/01	1982213
BLACKWOOD	ROSEMARIE	MIS	82/04/01	1982091
CROSS	BARBARA	MIS	81/11/02	2081306
GREENSPAN	MARY	MIS	82/04/01	1982091
IRVING	JOAN	PRODUCTION	82/01/04	1982004
JONES	DIANE	MIS	82/05/01	1982121
MCCOY	JOHN	MIS	81/07/01	2081182
MCKNIGHT	ROGER	PRODUCTION	82/02/02	1982033
ROMANS	ANTHONY	PRODUCTION	82/07/01	1982182
SMITH	MARY	MIS	81/07/01	2081182
	RICHARD	PRODUCTION	82/01/04	1982004
STEVENS	ALFRED	PRODUCTION	80/06/02	2080154

Additional Support for Cross-Century Dates

The following features apply to the use of dates in your applications.

Default Date Display Format

The default date display format is MM/DD/CCYY, where MM is the month; DD is the day of the month; CC is the first two digits of a 4-digit year, indicating the century; and YY is the last two digits of a 4-digit year.

For example:

02/11/1999

For a table that fully describes the display of a date based on the specified format and user input, see the *Describing Data* manual.

Date Display Options

The following date display options are available:

- You can display a row of data, even though it contains an invalid date field, using the command SET ALLOWCVTERR. The invalid date field is returned as the base date or as blanks, depending on other settings. For details, see your documentation on the SET command. This feature applies to non-FOCUS data sources when converting from the way data is stored (ACTUAL attribute) to the way it is formatted (FORMAT or USAGE attribute).
- If a date format field contains the value zero (0), you can display its base date, using the command SET DATEDISPLAY = ON. By default, the value zero in a date format field such as YYMD is returned as a blank. For details, see Chapter 1, *Customizing Your Environment*.
- You can display the current date with a 4-digit year using the Dialogue Manager system variables &YYMD, &MDYY, and &DMYY. The system variable &DATEfmt displays the current date as specified by the value of *fmt*, which is a combination of allowable date options, including a 4-digit year (for example, &DATEYYMD). For details, see Chapter 3, *Managing an Application With Dialogue Manager*.

System Date Masking

You can temporarily alter the system date for application testing and debugging, using the command SET TESTDATE. With this feature, you can simulate clock settings beyond the year 1999 to determine the way your program will behave. For details, see Chapter 1, *Customizing Your Environment*.

Date Functions

The date functions supplied with your software work across centuries. Many of them facilitate date manipulation. For details on date functions, see the *Using Functions* manual.

Date Conversion

You can convert a legacy date to a date format in a FOCUS data source using the option DATE NEW on the REBUILD command. For details, see the *Maintaining Databases* manual.

Century and Threshold Information

The ALL option, in conjunction with the HOLD option, on the CHECK FILE command includes file-level and field-level default century and year thresholds as specified in a Master File. For details, see the *Describing Data* manual.

Date Time Stamp

The year in the time stamp for a FOCUS data source is physically written to page one of the file in the format CCYY.

CHAPTER 7

Euro Currency Conversion

Topics:

- Integrating the Euro Currency
- Converting Currencies
- Creating the Currency Data Source
- Identifying Fields That Contain Currency Data
- Activating the Currency Data Source
- Processing Currency Data
- Querying the Currency Data Source in Effect

This topic describes how to create and use a currency data source to convert to and from the new euro currency.

Integrating the Euro Currency

With the introduction of the euro currency, businesses need to maintain books in two currencies, add new fields to their data source designs, and perform new types of currency conversions. You can perform currency conversions according to the rules specified by the European Union. To do this:

1. Create a currency data source with the currency IDs and exchange rates you will use. See *Creating the Currency Data Source* on page 7-4.
2. Identify fields in your data sources that represent currency data. See *Identifying Fields That Contain Currency Data* on page 7-6.
3. Activate your currency data source. See *Activating the Currency Data Source* on page 7-8.
4. Perform currency conversions. See *Processing Currency Data* on page 7-9.

Note: As the euro symbol becomes available to operating systems, Information Builders will support it.

Converting Currencies

Although the euro was introduced in 11 countries of the European Union on January 1, 1999, it will not immediately replace local currencies in those countries. During the transition period from 1999 to 2002, both traditional currencies and the euro will be used simultaneously for accounting purposes and non-cash transactions in each participating country. Euro cash will be introduced on January 1, 2002, and by July 1, 2002 the traditional currencies will no longer be legal tender.

In 1998, the European Union set fixed exchange rates between the euro and the traditional national currency in each of the 11 adopting member nations. Although 11 or more currencies in the European Union will be converting to the euro, more than 100 currencies have a recognized status worldwide. In addition, you may need to define custom currencies for some applications.

While the exchange rates within Euroland will remain fixed, exchange rates between the euro and non-euro countries will continue to vary freely and, in fact, several rates may be in use at one time (for example, actual and budgeted rates).

You identify your currency codes and rates by creating a currency data source. For more information see *Creating the Currency Data Source* on page 7-4.

Reference

Currency Conversion Rules

The European Union has established the following rules for currency conversions:

- The exchange rate must be specified as a decimal value, r , with six significant digits. This rate will establish the following relationship between the euro and the particular national currency:
 $1 \text{ euro} = r \text{ national units}$
- To convert from the euro to the national unit, multiply by r and round the result to two decimal places.
- To convert from the national currency to the euro, divide by r and round the result to two decimal places.
- To convert from one national currency to another, first convert from one national unit to the euro, rounding the result to at least three decimal places (your application rounds to exactly three decimal places). Then convert from the euro to the second national unit, rounding the result to two decimal places. This two-step conversion process is *triangulation*.

Example

Performing Triangulation

The following example illustrates triangulation. In this case, 10 US dollars (USD) are converted to French francs (FRF). The exchange rate for USD to euros (EUR) is 1.17249. The exchange rate for FRF to euros is 6.55957.

- The 10 USD are converted to EUR by dividing the 10 USD by the EUR exchange rate of 0.8840:
$$\text{EUR} = 10 / 0.8840$$

This results in 11.3122 euros.
- The euros are converted to FRF by multiplying the above result by the exchange rate of FRF for euros (6.55957):
$$\text{FRF} = 11.3122 * 6.55957$$

The result is 74.26. FRF. This means 74.26 FRF are equivalent to 10 USD.

Creating the Currency Data Source

For each type of currency you need, you must supply the following values in your currency data source:

- A three-character code to identify the currency, such as USD for US dollars or BEF for Belgian francs. (For a partial list of recognized currency codes, see *Sample Currency Codes* on page 7-5.)
- One or more exchange rates for the currency.

There is no limit to the number of currencies you can add to your currency data source, and the currencies you can define are not limited to official currencies and. Therefore, the currency data source can be fully customized for your applications.

The currency data source can be any type of data source your application can access (for example, FOCUS, FIX, DB2, or VSAM). The currency Master File must have one field that identifies each currency ID you will use and one or more fields to specify the exchange rates.

We strongly recommend that you create a separate data source for the currency data rather than adding the currency fields to another data source. A separate currency data source enhances performance and minimizes resource utilization because the currency data source is loaded into memory before you perform currency conversions.

Syntax

How to Create a Currency Data Source

```
FILE = name, SUFFIX = suffix,$
FIELD = CURRENCY_ID,, FORMAT = A3, [ACTUAL = A3 ,]$
FIELD = rate_1,, FORMAT = {D12.6|numeric_format1}, [ACTUAL = A12,]$
.
.
.
FIELD = rate_n,, FORMAT = {D12.6|numeric_formatn}, [ACTUAL = A12,]$
```

where:

name

Is the name of the currency data source.

suffix

Is the suffix of the currency data source. The currency data source can be any type of data source your application can access.

CURRENCY_ID

Is the required field name. The values stored in this field are the three-character codes that identify each currency, such as USD for U.S. dollars. Each currency ID can be a universally recognized code or a user-defined code.

Note: The code EUR is automatically recognized; you should *not* store this code in your currency data source. See *Sample Currency Codes* on page 7-5 for a list of common currency codes.

rate_1...rate_n

Are types of rates (such as BUDGET, FASB, ACTUAL) to be used in currency conversions. Each rate is the number of national units that represent one euro.

numeric_format1...numeric_formatn

Are the display formats for the exchange rates. Each format must be numeric. The recommended format, D12.6, ensures that the rate is expressed with six significant digits as required by the European Union conversion rules. Do not use Integer format (I).

ACTUAL An

Is required only for non-FOCUS data sources.

Note: The maximum number of fields in the currency data source must not exceed 255 (that is, the CURRENCY_ID field plus 254 currency conversion fields).

Reference

Sample Currency Codes

On December 31, 1998, Euroland set exchange rates between the euro and other currencies. Countries included in Euroland as of that date are marked with an asterisk (*). Their rates are fixed and will not change; the rates for other countries change over time.

Country	Currency Code	Rate
Austria*	ATS	13.7603
Belgium*	BEF	40.3399
Canada	CAD	1.3765
Denmark	DKK	7.4624
European Union	EUR	1
Finland*	FIM	5.94573
France*	FRF	6.55957
Germany*	DEM	1.95583
Greece*	GRD	340.750
Ireland*	IEP	0.787564
Italy*	ITL	1936.27
Japan	JPY	109.80
Luxembourg*	LUF	40.3399
Netherlands*	NLG	2.20371
Norway	NOK	8.0665
Portugal*	PTE	200.482
Spain*	ESP	166.386
Sweden	SEK	9.0125
Switzerland	CHF	1.5224
UK	GBP	0.61790
USA	USD	0.8840

Example**Specifying Currency Codes and Rates in a Master File**

The following Master File for a comma-delimited currency data source specifies two rates for each currency, ACTUAL and BUDGET:

```
FILE = CURRCODE, SUFFIX = COM,$  
FIELD = CURRENCY_ID,, FORMAT = A3, ACTUAL = A3 ,$  
FIELD = ACTUAL, ALIAS =, FORMAT = D12.6, ACTUAL = A12 ,$  
FIELD = BUDGET, ALIAS =, FORMAT = D12.6, ACTUAL = A12 , $
```

The following is sample data for the currency data source defined by this Master File:

```
FRF, 6.55957, 6.50000,$  
USD, 1.17249, 1.20000,$  
BEF, 40.3399, 41.00000,$
```

Identifying Fields That Contain Currency Data

Once you have created your currency data source, you must identify the fields in your data sources that represent currency values. To designate a field as a currency-denominated value (a value that represents a number of units in a specific type of currency) add the CURRENCY attribute to one of the following:

- The FIELD specification in the Master File.
- The left side of a DEFINE or COMPUTE.

Syntax**How to Identify a Currency Value**

Use the following syntax to identify a currency-denominated value

In a Master File

```
FIELD = currfield,, FORMAT = numeric_format, ..., CURR = {curr_id|codefield} , $
```

In a DEFINE in the Master File

```
DEFINE currfield/numeric_format CURR curr_id = expression ; $
```

In a DEFINE FILE command

```
DEFINE FILE filename  
currfield/numeric_format CURR curr_id = expression ;  
END
```


In a COMPUTE command

```
COMPUTE currfield/numeric_format CURR curr_id = expression ;
```

where:

currfield

Is the name of the currency-denominated field.

numeric_format

Is a numeric format. Depending on the currency denomination involved, the recommended number of decimal places is either two or zero. Do not use I or F format.

CURR

Indicates that the field value represents a currency-denominated value. CURR is an abbreviation of CURRENCY, which is the full attribute name.

curr_id

Is the three-character currency ID associated with the field. In order to perform currency conversions, this ID must either be the value EUR or match a CURRENCY_ID value in your currency data source.

codefield

Is the name of a field, qualified if necessary, that contains the currency ID associated with *currfield*. The code field should have format A3 or longer and is interpreted as containing the currency ID value in its first three bytes. For example:

```
FIELD = PRICE,, FORMAT = P12.2C, ..., CURR = TABLE.FLD1,$
.
.
.
FIELD = FLD1,, FORMAT = A3, ..., $
```

The field named FLD1 contains the currency ID for the field named PRICE.

filename

Is the name of the file for which this field is defined.

expression

Is a valid expression.

Example

Identifying a Currency-Denominated Field

The following Master File contains the description of a field named PRICE that is denominated in U. S. dollars.

```
FILE=CURRDATA,SUFFIX=COM,$
FIELD=PRICE, FORMAT=P17.2 , ACTUAL=A5, CURR=USD,$
.
.
.
```

Activating the Currency Data Source

Before you can perform currency conversions, you must specify the currency data source by setting the EUROFILE parameter with the SET command. By default, the EUROFILE parameter is not set.

The SET command can be issued at the FOCUS command prompt, in a procedure, or in any supported profile. It cannot be set within a report request.

Once a data source is activated, you can access a different currency data source by re-issuing the SET command.

Note: The EUROFILE parameter must be set alone. For example, appending an additional SET parameter will cause the additional parameter setting to be lost.

Syntax

How to Activate Your Currency Data Source

```
SET EUROFILE = {ddname|OFF}
```

where:

ddname

Is the name of the Master File for the currency data source. The *ddname* must refer to a data source known to and accessible by your application in read-only mode.

OFF

Deactivates the currency data source and removes it from memory.

Reference

EUROFILE Error Messages and Notes

- Issuing the SET EUROFILE command when the currency data source Master File does not exist generates the following error message:
(FOC205) THE DESCRIPTION CANNOT BE FOUND FOR FILE NAMED: *ddname*
- Issuing the SET EUROFILE command when the currency Master File specifies a FOCUS data source and the associated FOCUS data source does not exist generates the following error message:
(FOC036) NO DATA FOUND FOR THE FOCUS FILE NAMED: *name*

Note for Pooled Table users: The SET EUROFILE command creates a pool boundary.

Processing Currency Data

After you have created your currency data source, identified the currency-denominated fields in your data sources, and activated your currency data source, you can perform currency conversions.

Each currency ID in your currency data source generates a virtual conversion function whose name is the same as its currency ID. For example, if you added BEF to your currency data source, a virtual BEF currency conversion function will be generated.

The euro function, EUR, is supplied automatically with your application. You do not need to add the EUR currency ID to your currency data source.

The result of a conversion is calculated with very high precision, 31 to 36 significant digits, depending on platform. The precision of the final result is always rounded to two decimal places. In order to display the result to the proper precision, its format must allow at least two decimal places.

Syntax

How to Process Currency Data

In a procedure:

```
DEFINE FILE filename
```

```
result/format [CURR curr_id] = curr_id(infield, rate1 [,rate2]);
```

```
END
```

or

```
COMPUTE result/format [CURR curr_id] = curr_id(infield, rate1 [,rate2]);
```

In a Master File:

```
DEFINE result/format [CURR curr_id] = curr_id(infield, rate1 [,rate2]);$
```

where:

filename

Is the name of the file for which this field is defined.

result

Is the converted currency value.

format

Is a numeric format. Depending on the currency denomination involved, the recommended number of decimal places is either two or zero. The result will always be rounded to two decimal places, which will display if the format allows at least two decimal places. Do not use an Integer or Floating Point format.

curr_id

Is the currency ID of the result field. This ID must be the value EUR or match a currency ID in your currency data source; any other value generates the following message:

(FOC263) EXTERNAL FUNCTION OR LOAD MODULE NOT FOUND: *curr_id*

Note: The CURR attribute on the left side of the DEFINE or COMPUTE identifies the result field as a currency-denominated value which can be passed as an argument to a currency function in subsequent currency calculations. Adding this attribute to the left side of the DEFINE or COMPUTE does not invoke any format or value conversion on the calculated result.

infield

Is a currency-denominated value. This input value will be converted from its original currency to the *curr_id* denomination. If the *infield* and *result* currencies are the same, no calculation is performed and the *result* value is the same as the *infield* value.

rate1

Is the name of a rate field from the currency data source. The *infield* value is divided by its currency's *rate1* value to produce the equivalent number of euros.

If *rate2* is not specified in the currency calculation and triangulation is required, this intermediate result is then multiplied by the *result* currency's *rate1* value to complete the conversion.

In certain cases, you may need to provide different rates for special purposes. In these situations you can specify any field or numeric constant for *rate1* as long as it indicates the number of units of the *infield* currency denomination that equals one euro.

rate2

Is the name of a rate field from the currency data source. This argument is only used for those cases of triangulation in which you need to specify different rate fields for the *infield* and *result* currencies. It is ignored if the euro is one of the currencies involved in the calculation.

The number of euros that was derived using *rate1* is multiplied by the *result* currency's *rate2* value to complete the conversion.

In certain cases, you may need to provide different rates for special purposes. In these situations you can specify any field or numeric constant for *rate2* as long as it indicates the number of units of the *result* currency denomination that equals one euro.

Reference**Currency Calculation Error Messages**

Issuing a report request against a Master File that specifies a currency code not listed in the active currency data source generates the following message:

```
(FOC1911) CURRENCY IN FILE DESCRIPTION NOT FOUND IN DATA
```

A syntax error or undefined field name in a currency conversion expression generates the following message:

```
(FOC1912) ERROR IN PARSING CURRENCY STATEMENT
```

Example**Using the Currency Conversion Function**

Assume that the currency data source contains the currency IDs USD and BEF, and that PRICE is denominated in Belgian francs as follows:

```
FIELD = PRICE, ALIAS=, FORMAT = P17.2, CURR=BEF,$
```

- The following example converts PRICE to euros and stores the result in PRICE2 using the BUDGET conversion rate for the BEF currency ID:

```
COMPUTE PRICE2/P17.2 CURR EUR = EUR(PRICE, BUDGET);
```

- This example converts PRICE from Belgian francs to US dollars using the triangulation rule:

```
DEFINE PRICE3/P17.2 CURR USD = USD(PRICE, ACTUAL);$
```

First PRICE is divided by the ACTUAL rate for Belgian francs to derive the number of euros rounded to three decimal places. Then this intermediate value is multiplied by the ACTUAL rate for US dollars and rounded to two decimal places.

- The following example uses a numeric constant for the conversion rate:

```
DEFINE PRICE4/P17.2 CURR EUR = EUR(PRICE,5);$
```

- The next example uses the ACTUAL rate for Belgian francs in the division and the BUDGET rate for US dollars in the multiplication:

```
DEFINE PRICE5/P17.2 CURR USD = USD(PRICE, ACTUAL, BUDGET);$
```

Example**Converting U.S. Dollars to Euros, French Francs, and Belgian Francs****First**

Create a currency data source that identifies the currency and one or more exchange rates. (See *Creating the Currency Data Source* on page 7-4 for details.) The following sample data source is named CURRCODE:

```
FILE = CURRCODE, SUFFIX = COM,$
FIELD = CURRENCY_ID,, FORMAT = A3, ACTUAL = A3 ,$
FIELD = ACTUAL, ALIAS =, FORMAT = D12.6, ACTUAL = A12 ,$
FIELD = BUDGET, ALIAS =, FORMAT = D12.6, ACTUAL = A12 , $
```

Second

Create a data source that contains the values to be converted. (See *Identifying Fields That Contain Currency Data* on page 7-6 for details.) The following sample data source is named CURRDATA:

```
FILE=CURRDATA,SUFFIX=COM,$
FIELD=PRICE, FORMAT=P17.2 , ACTUAL=A5, CURR=USD,$
```

Third

Create a request that uses the currency data source to convert the currency values contained in the data source containing these values. The following procedure converts PRICE to euros, French francs, and Belgian francs. The numbers on the left correspond to the notes explaining the code.

```
      - * THE FOLLOWING FILEDEFS ARE FOR RUNNING UNDER CMS
1.  - * CMS FILEDEF CURRCODE DISK CURRCODE TEXT A
2.  - * CMS FILEDEF CURRDATA DISK CURRDATA TEXT A

      - * THE FOLLOWING ALLOCATIONS ARE FOR RUNNING UNDER MVS
1.  - * DYNAM ALLOC FILE CURRCODE DA USER1.FOCEXEC.DATA(CURRCODE) SHR REU
2.  - * DYNAM ALLOC FILE CURRDATA DA USER1.FOCEXEC.DATA(CURRDATA) SHR REU

3.  SET EUROFILE = CURRCODE

      DEFINE FILE CURRDATA
4.  PRICEUR/P17.2 CURR EUR = EUR(PRICE, ACTUAL);
      END

      TABLE FILE CURRDATA
      PRINT PRICE PRICEUR AND COMPUTE
5.  PRICEFRF/P17.2 CURR FRF = FRF(PRICE, ACTUAL);
      PRICEBEF/P17.2 CURR BEF = BEF(PRICE, ACTUAL);
      END
```

The report request executes as follows:

1. The FILEDEF or DYNAM command informs the operating system of the location of the CURRCODE data source.
2. The FILEDEF command informs the operating system of the location of the CURRDATA data source.
3. The SET command specifies the currency data source as CURRCODE.
4. This line calls the EUR function, which converts U. S. dollars to euros.
5. The next two lines are the conversion functions that convert euros into their equivalent in French and Belgian Francs.

The output is:

PRICE	PRICEEUR	PRICEFRF	PRICEBEF
-----	-----	-----	-----
5.00	4.26	27.97	172.01
6.00	5.12	33.57	206.42
40.00	34.12	223.78	1376.20
10.00	8.53	55.95	344.06

Note: You cannot use the derived euro value PRICEEUR in a conversion from USD to BEF. PRICEEUR has two decimal places (P17.2), not three, as the triangulation rules require. Therefore, PRICEEUR yields the following inaccurate result (see PRICEBEF above) and is not valid as the intermediate value in a currency conversion that requires triangulation:

```
COMPUTE PRICENEW/P17.2 CURR BEF = BEF(PRICEEUR, ACTUAL);
```

```
PRICENEW
-----
  171.85
  206.54
 1376.40
  344.10
```

Querying the Currency Data Source in Effect

You can issue a query to determine what currency data source is in effect. To do this, issue the ? SET ALL query command or the ? EUROFILE query command.

Syntax **How to Determine the Currency Data Source in Effect**

```
? SET EUROFILE
```

Example **Determining the Currency Data Source in Effect**

Assume the currency data source is named CURRCODE.

If you issue the following commands:

```
SET EUROFILE = CURRCODE
```

```
? SET EUROFILE
```

FOCUS returns the following response:

```
EUROFILE                      CURRCODE
```

CHAPTER 8

Designing Windows With Window Painter

Topics:

- Introduction
- Window Files and Windows
- Integrating Windows and the FOCEXEC
- Tutorial: A Menu-Driven Application
- Window Painter Screens
- Transferring Window Files

This topic describes how to create FOCUS menus and windows that work with FOCEXECs.

Introduction

FOCUS Window Painter is a tool that helps you design and create your own menus and screens for attractive and easy-to-use applications.

Many window types and features are available. You can implement horizontal menus and multi-input windows as part of your FOCUS application. Horizontal menus can also have pulldown menus associated with each menu item.

You can perform a string search in an active window by entering any pattern followed by a blank and then pressing Enter. Within the pattern:

- An asterisk (*) is a multiple character wildcard.
- A question mark (?) is a single character wildcard.
- An equal sign (=) repeats the last string.

FOCUS tries to locate the line matching the pattern starting from the line following the current line. The search concludes at the line preceding the current line. If no match is found, a beep sounds and the cursor remains at the current position.

The windows you can design with FOCUS Window Painter look just like the menus and screens you see in the FOCUS Talk Technologies, such as TableTalk and PlotTalk, but you can customize them to fit your application. You can design user-friendly menus and can display convenient and eye-catching instructions onscreen.

FOCUS Window Painter itself guides you step by step, using windows like those you will be creating.

On the windows you create, you can prompt users to:

- Select menu items from a list.
- Enter data.
- Select from automatically generated lists of available files and field names.
- Register a choice by pressing a function key.

You can also simply display explanations and instructions.

Window Painter is flexible enough to design the many different types of windows you might need for any application you can write with FOCUS.

You can also upload window files from FOCUS running in one operating environment, such as PC/FOCUS, and edit them using Window Painter for use on another operating environment such as MVS or CMS.

How Do Window Applications Work?

Window Painter stores the windows you design in window files. Window files work in conjunction with FOCEXEC procedures that use Dialogue Manager.

There are two major parts in any window application, each of which is a step for the developer:

- The windows, created with Window Painter, which users will see.
- The Dialogue Manager FOCEXEC.

You can invoke Window Painter to create and edit windows by typing

`WINDOW [PAINT]`

at the FOCUS prompt, and pressing Enter.

You can invoke the Window facility in your FOCEXEC by including the Dialogue Manager command `-WINDOW` in the FOCEXEC. The `-WINDOW` command provides the name of the window file, and the name of the individual window that should be displayed first. When the `-WINDOW` command is executed by Dialogue Manager, control in the FOCEXEC passes to the Window facility.

The user is moved through the window file by goto values. A goto value tells the Window facility which window to display next.

You specify goto values when creating the windows with Window Painter. When your window is a menu with several items, you may assign a different goto value for each menu item, so that the next window depends on the user's selection.

When you create the windows, you also specify return values. As with goto values, you may assign a different return value to each item on a menu. Return values are collected as the user moves through the windows, and are substituted for "amper variables" which can be used later in the window file or in the FOCEXEC when control passes back. (Amper variables are Dialogue Manager variables of the format `&variablename`.)

When the selected value is inserted in the FOCEXEC, you may test it with a Dialogue Manager `IF...THEN` command and branch accordingly to a label in the FOCEXEC. In this way, you move the user through a series of windows, collecting return values for amper variables, using only one command in your FOCEXEC.

You can use windows to collect amper variable values in place of any other method of prompting available through Dialogue Manager.

For a complete discussion of the Dialogue Manager facility, see Chapter 3, *Managing Applications With Dialogue Manager*. For details of integrating a FOCEXEC with the Window facility using return and goto values, see *Integrating Windows and the FOCEXEC* on page 8-21.

Window Files and Windows

Windows—that is, menus and screens—are stored in window files. Windows are included in a specified window file as you create and save them during a Window Painter session.

- In CMS, window files have file type FMU, and are created and updated on the A disk automatically by Window Painter.
- In MVS, window files are contained in a partitioned data set (PDS) allocated to ddname FMU. Before any window files can be created, a PDS must be created and ddname FMU must be allocated to it.

Note, however, that creating a PDS is not necessary if you are creating window files to be used only in the current FOCUS session: Window Painter will temporarily allocate the PDS. For a full description of allocation requirements, see the appropriate *Guide to Operations* topic in the *FOCUS Overview and Operating Environments* manual.

A window file can contain a maximum of 384 windows, and a number of windows may be displayed on the screen at once. All the windows in a single application may be stored together in one window file, or you may create separate window files for different parts of the application such as Help Windows.

You can make an application more attractive by presenting menus in windows containing titles and other design elements, and can make an application easier to use by displaying function key definitions or other useful information.

Types of Windows You Can Create

Window Painter creates 10 different types of windows, each with its own special uses:

- Vertical menus
- Horizontal menus
- Text input windows
- Text display windows
- File names windows
- Field names windows
- File contents windows
- Return value display windows
- Execution windows
- Multi-input windows

These windows are described in the following topics.

Vertical Menu

This is a *vertical* menu:

```

+-----+
|Select a report and press ENTER:|
+-----+
| Accounts Payable               |
| Accounts Receivable           |
| Salary Information             |
| Create a New Report            |
+-----+

```

A menu is a window that lets users select an option from a list. These options are called menu items. A vertical menu lists its menu items one below the other. A user can select an item by moving the cursor down the list with the arrow keys and pressing Enter when the cursor is on the line of the desired item. A user can select more than one item if the window includes the Multi-Select option, which is part of the Window Options Menu. Help information can be specified for each item in the menu by using the menu-item help feature of help windows. For additional information on Multi-Select and Help windows, see *Window Options Menu* on page 8-61.

Horizontal Menu

This is a *horizontal* menu:

```

+-----+
| Reporting   Ad hoc   Maintenance   Quit   |
+-----+

```

A horizontal menu displays its menu items on a line, from left to right. You select an item by using PF11 or the Tab key to move right and PF10 or Shift+Tab to move left across the line, and pressing Enter when the cursor is at the desired item. You can also select an item by employing the search techniques available for FOCUS windows. (Search techniques are not available with pulldown windows).

If you use PF11 at the last item on the menu, the cursor moves to the first item on the menu. If you use PF10 at the first item on the menu, the cursor moves to the last item on the menu, unless there is another screen to scroll to.

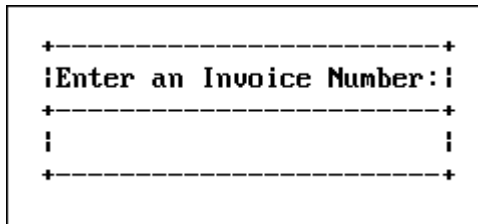
An application can display an associated pulldown menu for an item on a horizontal menu when the cursor is on that item. Choose the pulldown option from the Window Options menu as discussed in *Creating Windows* on page 8-14. An option to display descriptive text above or below the horizontal menu is also available from the Window Options menu.

You can assign any return value to each item on the menu. When you select a menu item, the corresponding return value is collected.

In a horizontal or vertical menu, you can assign a goto value to each menu item.

Text Input Windows

This is a *text input* window:



```

+-----+
!Enter an Invoice Number:!
+-----+
!                               !
+-----+

```

Amper variables can be used in a Windows application. A text input window prompts the user to supply information needed in a FOCEXEC. It is also possible to display an existing value to be edited. Each text input window accepts one line of input up to 76 characters long. You assign the length and format of the field when you create the window. Additional information about creating a text input window is found in *Window Creation Menu* on page 8-57.

Text Display Windows

This is a *text display* window:

```

+-----+
|Instructions for printing:|
+-----+
|
| Press ALT-7 if you wish |
| to generate an OFFLINE |
| report.                 |
|
| Press ALT-8 if you wish |
| to generate an ONLINE  |
| report.                 |
|
+-----+

```

A text display window lets you present information such as instructions or messages. No selections can be made from a text display window, and no data can be entered in it.

File Names Windows

This is a *file names* window:

```

+-----+
|Select the report you wish to generate and press ENTER:|
+-----+
| (MORE) |
| SALARY FOCEXEC B1 |
| ACCTS  FOCEXEC B1 |
| BILLS  FOCEXEC B1 |
| BUDGET FOCEXEC B1 |
| (MORE) |
+-----+

```

A file names window presents a list of names of up to 409 files (in CMS) or 1023 PDS members (in MVS). The user can select one of these names by moving the cursor and pressing Enter when the cursor is on the line of the desired file name. You can specify selection criteria for the displayed file names when the window is created. A user can select more than one file if the window includes the Multi-Select option, which is available on the Window Options Menu.

Note that the maximum number of file (or member) names which can be displayed decreases as the width of the window increases. Narrower windows can display a greater number of names.

Field Names Windows

This is a *field names* window:

```
+-----+
|Select the field you wish to sort on and press ENTER:|
+-----+
| EMP_ID                                           |
| LAST_NAME                                        |
| FIRST_NAME                                       |
| HIRE_DATE                                        |
| DEPARTMENT                                       |
| CURR_SAL                                         |
|+(MORE)-----+
+-----+
```

A field names window presents a list of all field names from a Master File; the user can select one by moving the cursor and pressing Enter when the cursor is on the line of the desired field name. A user can select more than one field if the window includes the Multi-Select option, which is available on the Window Options Menu.

You can use a field names window as the next step after a file names window. That way, you can present a selection of files first, followed by the fields in a selected file.

The field names will be qualified when duplicates exist. You can use PF10 and PF11 to scroll left and right if a field name exceeds the maximum number of characters allowed on a line in a data field window.

Use PF6 as a three-way toggle to sort the fields in one of the following ways:

1. Display field names in the order in which they appear in the Master File.
2. Display field names in alphabetical order.
3. Display the fully qualified field names in the order in which they appear in the Master File.

File Contents Windows

This is a *file contents* window:

```

+-----+
|Select the record you want to display and press ENTER:|
+-----+
| STAMFORD          S  14B                               |
| NEW YORK          U  14Z                               |
| UNIONDALE         R  77F                               |
| NEWARK            U   K1                               |
+-----+

```

The file contents window displays the contents of a file. There is no limit on the size of a file contents window. The user can select a line of contents by moving the cursor to it and pressing Enter. Each line can be up to 77 characters long. A user can select more than one line if the window includes the Multi-Select option, which is described as part of the Window Options Menu in *Window Options Menu* on page 8-61.

- In CMS, the contents of any file (except as noted below) can be displayed. You will be prompted for the file name and file type.
- In MVS, the contents of any member of a PDS (except as noted below) can be displayed. Sequential files can also be displayed in TSO. You will be prompted for a file name (the ddname) and a file type (the member name). This information should be entered as “member name ddname.”

Note: You cannot display a file with unprintable characters in a file contents window. This includes files such as FOCUS files, HOLD files, SAVB files, FOCCOMP files, and encrypted files.

Return Value Display Windows

This is a *return value display* window:

```

+-----+
|This is a sample Return Value Display window.|
+-----+
| TABLE FILE EMPLOYEE                               |
| PRINT EMP_ID FIRST_NAME LAST_NAME                 |
| END                                                |
+-----+

```

The return value display window displays amper variables that have been collected from other windows. No selections can be made from a return value display window, and no data can be entered into it.

Return value display windows are very useful for constructing a command (or any string of words or terms) by working through a series of windows. An example of this type of application is seen when you construct a TABLE request using TableTalk.

Each line of the return value display window is stored in a variable called *&windownamexx*, where *windowname* is the name of the window and *xx* is a line number.

Unless you use the Line-break option to place return values on separate lines, all collected return values are placed on the same line until the end of the line is reached. The length of the line is determined by the size of the window created. A description of the Line-break option on the Window Options Menu can be found in *Window Options Menu* on page 8-61.

Only one return value display window may be displayed at a time on the screen. It will collect a value from any active window (that is, a window from which a selection is being made or to which text is being entered, or an active text display window) if it is on that window's display list. A description of the Display lists option on the Window Options Menu can be found in *Window Options Menu* on page 8-61.

You can clear the collected values from a return value display window by including it on the hide list of a window that is being used. A description of the Hide lists option on the Window Options Menu can be found in *Window Options Menu* on page 8-61.

For a Multi-Select window, the return value display window gives the number of selections, not the values selected. The values can be retrieved by using the -WINDOW command with the GETHOLD option.

Execution Windows

This is an *execution* window:

```

+-----+
| -* This is a sample Execution window. |
| TABLE FILE EMPLOYEE |
| PRINT EMP_ID BY LAST_NAME |
| END |
| -RUN |
+-----+

Wind: EXECWIND Typ: Execution PF1=Help 2=Menu 4=Size 9=Move 10=Del 11=Add

```

The execution window contains FOCUS commands such as Dialogue Manager commands, and TABLE requests.

You can create an execution window by choosing its option on the Window Creation menu.

When this window is first displayed, it has a width of 77 characters, and no heading. You can place FOCUS commands within it. Note that the commands in an execution window appear just as you type them; commands are not automatically converted to uppercase.

The Window Painter Main Menu contains an option enabling you to run a window in order to see any return values collected. If you were to run (not execute) the execution window from the Window Painter Main Menu, you would see the execution window contents, then any windows called, and finally any return values collected by running the windows.

Note the following rules when using execution windows:

- When you GOTO an execution window, the contents of the window are executed. In all cases, execution begins at the top of the window.
- An execution window is not displayed when executed, although the commands it contains may generate a display.
- An execution window can use an amper variable as a goto value.
- An execution window clears the screen and the Return Value display window.

- Information Builders

```
+-----+
|      |
|      |
| Enter the following personnel information: |
|      |
| Name:          |
|      |
| Address:       |
| City:         , |
| Zip Code      - |
|      |
| Phone Number:  - - |
|      |
| Department:    |
|      |
+-----+
```

A multi-input window prompts you for information that will be used in the application. A multi-input window may include up to 50 input fields, each of which can be up to 76 characters long. You assign the length, name, and format of the field when you create the window.

Use the Tab key to move the cursor between the fields on a multi-input window.

You can supply help information for each field in a multi-input window by using the Help window option. For information on Help windows, see *Window Options Menu* on page 8-61.

For a multi-input window, the return value is the name of the input field occupied by the cursor when you pressed Enter or a function key. The name that you supply for each input field is assigned to an amper variable with the same name as the field (each input field has a unique name). The variable &WINDOWVALUE contains the value of the input field occupied by the cursor when you pressed Enter or a function key.

Use a unique name for each field on a multi-input window. To display the field names specified, use the Input Fields option on the Window Options menu.

Creating Windows

The process of creating windows begins with choosing the type of window you want to create from the Window Creation menu. Each type of window requires slightly different instructions. The tutorial in *Tutorial: A Menu-Driven Application* on page 8-29 describes how to create and implement text display window, vertical menu, and file names windows. This topic describes how to create horizontal menus (with or without associated pulldown menus) and multi-input windows.

Creating a Horizontal Menu

To create a horizontal menu, begin by placing the cursor at the Menu (horizontal) option on the Window Creation menu:

INSTRUCTIONS : Move cursor to selection and hit ENTER Use PF3 or PF12 to undo a selection Use PF1 for help	
<div style="border: 1px dashed black; padding: 5px; margin: 10px auto; width: 80%;"> +-----+ Select the window type: +-----+ Menu (vertical) Menu (horizontal) Text input Text display File names Field names File contents Return value display Execution window Multi-Input window +-----+ </div>	

You will be prompted to enter a name and brief description for the window, after which you will reach the creation screen. On this screen:

1. Move the cursor to the location in which you want the top left corner of the menu to be displayed. Press Enter.
2. Next, use the arrow keys to move the cursor down (enough spaces to leave a line for each item you want to display as a menu choice) and to the right (enough spaces to just fit the longest menu item). Press PF4. You will see two windows: one is for entering information and the other is the corresponding horizontal menu.
3. Enter the menu items in the window containing the cursor. Press the Enter key after each item; the item automatically appears on the horizontal menu.

The following is an example of a completed creation screen:

Vertical	Inputs	Lists	Execution	Misc	End
Vertical	Inputs	Lists	Execution	Misc	End
Wind: HORO Typ: Menu (horz) PF1=Help 2=Menu 4=Size 9=Move 10=Del 11=Add					

Once you have entered the items on your menu, there are several options you can select for each item. Move the cursor to any item and press PF2 to display the Window Options menu:

Exit this menu	
Goto value	
Return value	
FOCEXEC name	c Maintenance Quit
Heading	
Description	
Show a window	
Unshow a window	
Display list	
Hide list	
Popup (Off)	
Help window	
Line break	
Multi select (Off)	
Quit PF3	
Menu text	
Text line (x+1	
Pulldown (Off)	
Conceal option	
Switch window	

Position the cursor on any option you want to select and press Enter.

Two features available for horizontal menus are Menu text and Text line. Menu text is a line of text displayed when the cursor is on a menu item. The line on which the text is displayed is called the text line. You can position the text line one or two lines either above or below the horizontal menu.

The following example illustrates Menu text and Text line. When the cursor is positioned on Vertical in the example below, the following is displayed:

VERTICAL MENU TESTS					
Vertical	Inputs	Lists	Execution	Misc	End

In this example, the Menu text VERTICAL MENU TESTS is positioned at Text line x-1, one line above the menu. To place the Text line two lines above the Menu text, change x-1 to x-2. For Text lines below the menu text, use x+1 or x+2.

You can also select the Pulldown option for a horizontal menu. With this option, you can assign a pulldown menu to be displayed for a horizontal menu item whenever the cursor is positioned on that item.

Pulldown Menus

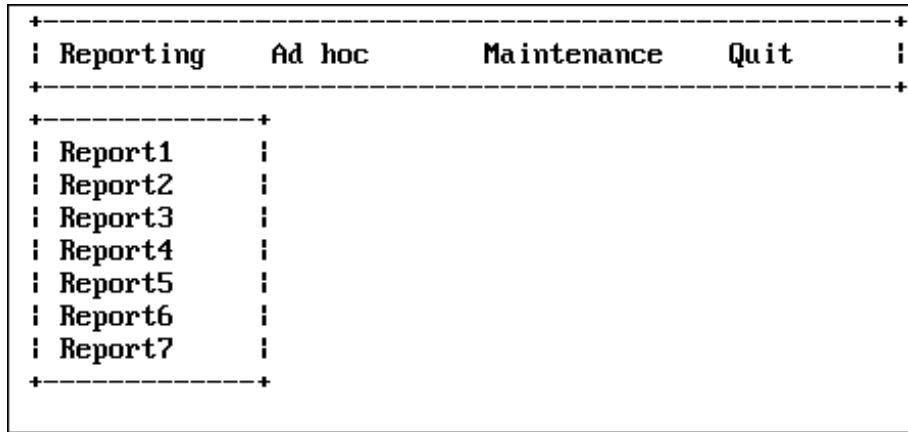
When you set the Pulldown option ON, you can display an associated pulldown menu for an item in a horizontal menu by positioning the cursor on that item. The default is OFF. To change the setting to ON, position the cursor on the Pulldown option and press Enter. Note that when Pulldown is set ON, Menu Text is automatically set OFF.

The associated pulldown menu must be a vertical menu. When creating the horizontal menu, you must assign a Goto value to point to the pulldown menu. To do so, position the cursor on the goto value, press Enter, and enter the name of the pulldown menu you want to display in the space provided:

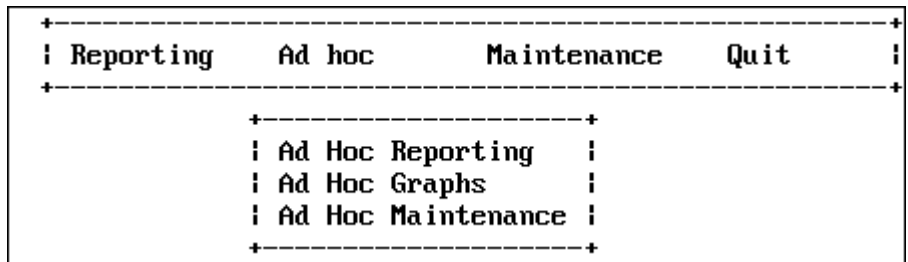
Reporting	Enter name of next window to go to.!
	Just 'Enter' for exit.
Reporting	rpts
Ad hoc	
Maintenance	
Quit	

You must create the vertical menu, rpts, as you would any other vertical menu. See *Tutorial: A Menu-Driven Application* on page 8-29 for examples.

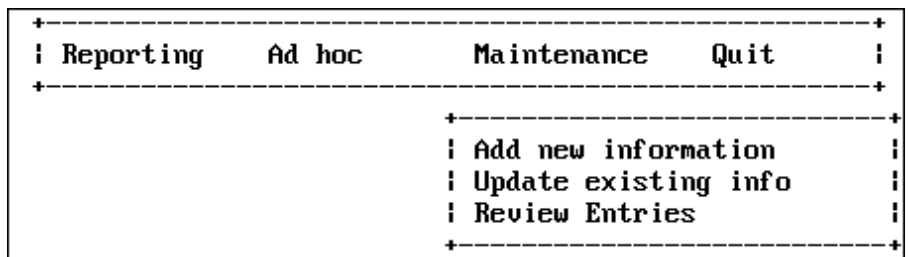
The following example shows a horizontal menu with the Reporting pulldown menu displayed:



The following screen shows the same menu with the Ad hoc pulldown menu displayed:



The following screen shows the same menu with the Maintenance pulldown menu displayed:



Note: To move from item to item in a horizontal menu, use PF10 and PF11.

Creating a Multi-Input Window

To create a multi-input window, begin by placing the cursor at the Multi-Input window option on the Window Creation menu and press Enter. You will then be prompted for a name, description and heading. Place the window on the screen and size it as desired.

+-----+	
	INSTRUCTIONS : Move cursor to selection and hit ENTER
	Use PF3 or PF12 to undo a selection
	Use PF1 for help
+-----+	
+-----+	
	Select the window type:
+-----+	
	Menu (vertical)
	Menu (horizontal)
	Text input
	Text display
	File names
	Field names
	File contents
	Return value display
	Execution window
	Multi-Input window
+-----+	

To place entries on the window:

1. Type the text for display.
2. Press PF6 at the point where the field begins.
3. Space along for the length of the field.
4. Press PF6 again to signify the end of the input area.
5. Enter name and information for the field.

The following example shows a multi-input window, with Name: entered as display text.

F O C U S W I N D O W P A I N T E R	
	INSTRUCTIONS : Move cursor to selection and hit ENTER
	Use PF3 or PF12 to undo a selection
	Use PF1 for help
+-----+	
	!Enter a description:
	!Sample file for Window Painter tutorial.
+-----+	

This is what the developer's screen looks like after several fields have been included in the multi-input window:

	!Enter the following personnel information:
+-----+	
	Name: XXXXXXXXXXXXXXXXXXXXXXXXXXXX
	Address: XXXXXXXXXXXXXXXXXXXX
	XXXXXXXXXX , XX
	Zip Code:XXXXX - XXXX
	Phone Number: XXX - XXX - XXXX
	Department: XXXXXXXXXXXX
+-----+	

Note: Text fields may be supplied without headings or instructions. For example, see the city and state portion of the address line.

This is how the window appears when run as part of the application:

+-----+-----+	
Enter the following personnel information:	
+-----+-----+	
Name:	
Address:	
Zip Code	- ,
Phone Number:	- -
Department:	
+-----+-----+	

The following screen shows what is returned from the window when it is run inside the Window Painter:

+-----+-----+	
Variable	Value
+-----+-----+	
&WINDOWNAME	MULTI
&WINDOWVALUE	
&MULTI	NAME
&NAME	
&STREET	
&CITY	
&STATE	
&ZIP1	
&ZIP4	
&AREA	
&EXCHANGE	
&NUMB	
&DEPARTMENT	
&PFKEY	ENTR
&RETCODE	0
+-----+-----+	

Note: To move from field to field in a multi-input window, use the Tab key.

Integrating Windows and the FOCEXEC

The windows you create with Window Painter are designed for you to use within an application FOCEXEC. This topic discusses how to integrate your windows into your FOCEXEC.

Syntax

The -WINDOW Command

To invoke the Window facility, insert the following Dialogue Manager command in your FOCEXEC

```
-WINDOW windowfile windowname [PFKEY|NOPFKEY] [GETHOLD] [BLANK|NOBLANK] [CLEAR|NOCLEAR]
```

where:

windowfile

Identifies the file in which the windows are stored. In CMS, this is a file name. The file must have a file type of FMU or TRF. In MVS, this is a member name. The member must belong to a PDS allocated to ddname FMU.

windowname

Optional. Identifies which window in the file to display first. Can be set in Window Painter or in first window displayed.

PFKEY/NOPFKEY

Enables (prevents) testing for function key values during window execution.

GETHOLD

Retrieves stored amper variables collected from a Multi-Select window. Does not cause window to be displayed.

BLANK

Clears all previously set amper variable values when the -WINDOW command is encountered. This is the default setting.

NOBLANK

No amper variable values are cleared when the -WINDOW command is encountered.

CLEAR

When FOCUS is being used with the Terminal Operator Environment (described in the *Overview and Operating Environments* manual), the -WINDOW command clears the screen before displaying the first window. The Terminal Operator Environment screen will be redisplayed when control is transferred from the Window facility back to the FOCEXEC. This is the default setting.

NOCLEAR

When FOCUS is being used with the Terminal Operator Environment, the window file's windows are displayed directly over the Terminal Operator Environment screens.

Note: NOBLANK is particularly important in applications that use more than one -WINDOW command.

Transferring Control in Window Applications

When the -WINDOW command is encountered, control in the FOCEXEC is transferred to the Window facility. Control remains with the Window facility until one of the following occurs:

- The user makes a selection for which you have assigned no goto value.
- The PFKEY option is in effect and the user presses a function key (the function key must be set to RETURN, HX, CANCEL, or END, as described in the *Testing Function Key Values* on page 8-26.)

Once control passes back to the FOCEXEC, control only returns to the Window facility if another WINDOW command is encountered.

Example

Window File in an Application FOCEXEC

This example shows an application FOCEXEC and a window file named REPORT which contains three windows: R1, R2, and R3.

The numbers at the left of the example refer to the flow of execution (that is, the order in which the commands and windows are executed).

```
1. -START
2. -WINDOW REPORT R1 PFKEY
   -*
3. -*Control is transferred from the above command
   -*to window R1 in window file REPORT.
   -*
4. -IF &PFKEY EQ PF05 GOTO LABEL1;
   -*
   -*Control returns to the above command from
   -*window R2 in window file REPORT.
   .
   .
   -LABEL1
5. -WINDOW REPORT R3
   -*
6. -*Control is transferred from the above command
   -*to window R3 in window file REPORT.
   -*
7. -IF &R3 EQ EXIT GOTO EXIT;
   -*
   -*Control returns to the above command from
   -*WINDOW R3 in window file REPORT.
   .
   .
   -EXIT
```

Note:

- At Step 3, the user selects an option from Window R1. This option's goto value is R2. Control is transferred to Window R2.
- The user presses a function key in Window R2. Control is transferred to the FOCEXEC, to the command following the -WINDOW command (Step 4).
- At Step 6, the user selects the option to exit; no goto value was set for that option. Control is transferred to the FOCEXEC, to the command following the -WINDOW command (Step 7).

The flow of control has certain implications for the design of your window applications:

- Any time you wish to pass control back to the FOCEXEC, the window or menu option must have no goto value, or else must prompt the user to press a function key (as described in *Testing Function Key Values* on page 8-26).
- At some point in the window session, control should return to the FOCEXEC so that the accumulated return values can be substituted for amper variables, and the variables then used in the FOCEXEC.
- Any time you wish to pass control from the FOCEXEC to the Window facility you must insert the -WINDOW command in the FOCEXEC.
- Note that it is not necessary to create a new window file for each -WINDOW command; you can simply enter the same file again at whatever window you wish.
- If you wish to test for a function key value in the middle of a series of windows, remember that pressing the function key automatically returns control to the FOCEXEC; an -IF test command should follow the -WINDOW command, and a second -WINDOW command should be placed after the -IF command to transfer control back to the window file.
- If you want to clear an existing set of variable values, you may do so by returning control to the FOCEXEC and executing another -WINDOW command with the BLANK option in effect.

To back up a step during window execution, the user may press the PF12 or PF24 keys. This will not cause control to pass to the FOCEXEC. However, you can force Dialogue Manager to return control to a FOCEXEC by a PF key setting as described in *Testing Function Key Values* on page 8-26.

Return Values

When the user responds to your window prompt by entering text, selecting an item from a menu, or pressing a function key, this response is the return value that fills in an amper variable in your FOCEXEC.

There are two ways in which amper variables are most commonly used in FOCEXECs:

- To collect values to plug into a FOCUS procedure such as a TABLE or GRAPH request so it can run.
- To test the value returned in a variable, and branch accordingly to a different part of the FOCEXEC or to another FOCEXEC.

The return value collected can be almost anything you desire: a character string, a number, the name of a file, a procedure name, or part of a FOCUS command.

A return value amper variable in the FOCEXEC has the same name as the window in which it is collected; that is:

&windowname

For example, the return value collected by the window MAIN supplies a value for the variable &MAIN.

- In vertical menu and horizontal menu windows, you assign any return value you wish to each item on the menu. If the user selects that option, that return value is collected.
- In text input windows, the return value is the text that the user types.
- In text display windows, you can assign one return value to the entire window. Unlike other return values, a text display window return value is collected as soon as control passes to the window, without the user needing to select anything.
- Return value display windows display return values collected from other types of windows. These return values can be displayed one per line, or several together on a single line. Although this type of window does not itself have a return value, each line has a corresponding amper variable (&windownamexx, where xx is the line number).
- For a multi-input window, the return value is the name of the input field on which the cursor is positioned when you press Enter or a PF key.
- In windows with the Multi-Select option, the return value is the number of items selected.
- In file names, field names, and file contents windows, the return value is, respectively, the file name, field name, or line of file contents that the user selects from the display.

Example

Return Value in a Menu-Driven Application

For example, assume that you have written a menu-driven application that enables a user to report from any one of a list of files. You have created a series of windows for this application, one of which is a file names window named FILE designed to collect a return value for &FILE. The window displays a list of all the user's files that meet certain file-identification criteria you specified when you created the window.

Your FOCEXEC contains these lines:

```
-START  
-WINDOW EXAMPLE FILE  
.  
.  
.  
TABLE FILE &FILE
```

When the user moves the cursor to SALES and presses ENTER, SALES is collected to be substituted for &FILE in the FOCEXEC:

```
TABLE FILE SALES
```

Goto Values

When you are creating your windows, you will also assign goto values telling the Window facility which window to display next. These values allow you to move the user through a series of windows, collecting return values for amper variables, without adding lines to your FOCEXEC.

- In vertical menu and horizontal menu windows, you assign a goto value for each menu item.
- In all other windows, you assign a single goto value.
- You can use an amper variable as a GOTO value.

As described in *Transferring Control in Window Applications* on page 8-22, if you assign no goto value to a menu option or window, control passes back to the FOCEXEC when the user selects that option or presses Enter at that window.

It is important not to confuse these goto values with the Dialogue Manager -GOTO command. The goto value points your application to a new window in the window file; the -GOTO command transfers control to a label in your FOCEXEC.

Returning From a Window to Its Caller

You can return from a window to its caller via the <ESCAPE> option. If you enter this string as the goto value of a window, control will return to the previous window upon completion of the current window (enter the right and left carets as part of the goto value).

Window System Variables

We have already discussed return values: these are specific to each window. Two other Window facility variables, &WINDOWNAME and &WINDOWVALUE, are specific to the -WINDOW session (not to each window) and receive their values when the Window facility passes control from a window file back to the FOCEXEC.

&WINDOWNAME

&WINDOWNAME is an amper variable containing the name of the last window that was displayed before the Window facility transferred control back to the FOCEXEC.

This variable can be used in many ways. For example, if the goto values/function key prompts in a window file allow a user to leave the window file from several different windows, you can test &WINDOWNAME in the FOCEXEC to determine which window the user was in last (and, therefore, which path the user navigated through the window file).

&WINDOWVALUE

&WINDOWVALUE is an amper variable containing the return value from the last window that was displayed before the Window facility transferred control back to the FOCEXEC. If the user selected a line for which no return value was set (for example, a blank line between two menu options in a vertical menu window), then &WINDOWVALUE will contain the line number of the line that was selected.

This variable can be used in many ways. For example, if the goto values/function key prompts allow a user to leave the window file from several different windows, and you need to know the return value of the last window the user was in before she or he left the file by pressing a function key, you can test &WINDOWVALUE.

Testing Function Key Values

If you wish to test for function key values, you must specify the PFKEY option on the -WINDOW command line. When the PFKEY option is set and a user presses a function key during window execution, the name of that key is stored in the amper variable &PFKEY.

For example, if the user presses PF1, the 4-character value of &PFKEY is PF01. If PF2, the value is PF02, and so forth. If the user presses Enter, the value is ENTR. The value of &PFKEY is reset each time the user presses a function key.

Note that if the PFKEY option is specified, the Window facility's default PF key actions are overridden by the general FOCUS PF key settings. This means that when you specify the PFKEY option, if you still want the standard Window facility PF key actions to be available to window users (for example, PF1 = HELP, PF3 = UNDO), you must use the SET command in your application FOCEXEC, followed by a -RUN command, to explicitly set those actions.

For example, if you specify the PFKEY option but you want to retain all of the Window facility's default PF key actions using the same PF keys, you need to include the following commands before the -WINDOW command in your application FOCEXEC:

```
SET PF01=HELP
SET PF03=UNDO
SET PF04=TOP
SET PF05=BOTTOM
SET PF06=SORT
SET PF07=BACKWARD
SET PF08=FORWARD
SET PF09=SELECT
SET PF10=LEFT
SET PF11=RIGHT
SET PF12=UNDO
-RUN
```

When you specify the PFKEY option, any PF key which you want to test for in the application FOCEXEC must be set to RETURN. (HX, CANCEL, and END also function as RETURN within the Window facility, and can be used in place of it.)

For example, if you design your application so that a user can press PF2 to choose an additional menu option, and therefore you want to test &PFKEY for the value PF02 in your application FOCEXEC, then you must include the following SET command before the -WINDOW command in your application FOCEXEC:

```
SET PF02=RETURN
```

The SET PF command is discussed in Chapter 1, *Customizing Your Environment*, and in the *Maintaining Databases* manual.

You can list the current general FOCUS PF key settings by issuing the ? PFKEY command. The ? PFKEY command is discussed in Chapter 2, *Querying Your Environment*.

The variable &PFKEY can be tested just like any other amper variable. Note that the name of the variable is always &PFKEY; it is not linked to a window name like other amper variables collected through windows.

You may test the PFKEY variable repeatedly throughout the FOCEXEC. Additional SET commands are not required.

One of the advantages of using the &PFKEY variable is that it enables you to collect two return values from a single menu. You might, for example, create a window called FILES, which prompts the user to enter the name of a file, then press PF7 to produce a graph or PF8 to produce a report. Both the file name as &FILES and the function key value as &PFKEY would be collected as return values.

It is always important to remember that pressing a function key will immediately return control to the FOCEXEC if that key was set to RETURN (or to HX, CANCEL, or END).

Note: If the cursor is on a menu that has a FOCEXEC associated with it, the FOCEXEC is executed and the GOTO value associated with the menu choice is assumed. The PFKEY is ignored.

In the example above, if the user presses a function key before typing the file name, the &FILES variable will not be collected. If the key was set to something other than RETURN, HX, CANCEL, or END, then the action it was set to is invoked, and control remains within the Window facility.

Executing a Window From the FOCUS Prompt

You can execute a window directly from the FOCUS command prompt.

Syntax

How to Execute a Window From the FOCUS Prompt

```
EX 'windowfile FMU' [windowname] [PFKEY|NOPFKEY] [BLANK|NOBLANK]  
[CLEAR|NOCLEAR]
```

where:

windowfile

Is the file containing the windows. It must have file type FMU, and appear within single quotation marks.

windowname

Identifies the first window to be executed. If a window name is not specified, FOCUS will execute the default start window, or the first window created.

PFKEY|NOPFKEY

Tells FOCUS you will (will not) be testing for function key values during execution.

BLANK

Clears previously set amper variables when the window is called. This is the default setting.

NOBLANK

Retains previously set amper variables.

CLEAR

When FOCUS is being used with the Terminal Operator Environment, the screen is cleared when the EX command is encountered. The Terminal Operator Environment screen is restored when the last window in the chain has been executed. This is the default setting.

NOCLEAR

When FOCUS is being used with the Terminal Operator Environment, the screen is not cleared when the EX command is encountered, and any windows are displayed within the Terminal Operator Environment screens.

For example, to execute the window MAIN in the window file REPORT, you could issue EX 'REPORT FMU' MAIN from the FOCUS command prompt, which is equivalent to issuing -WINDOW REPORT MAIN from Dialogue Manager.

Tutorial: A Menu-Driven Application

This tutorial describes a menu-driven system that clerical personnel can use to produce sales reports and graphs at your chain of retail stores. The system must fulfill three major requirements:

- **Ease of use.** Your system must let employees be productive without extensive training.
- **Functionality.** The system has to work properly with only a few steps.
- **Appearance.** There should be continuity between screens, and a general unity of design. The reports and graphs produced must be attractive and easy to read.

The application prompts the user to select reporting or creating a graph.

Then, the user may opt to execute an existing FOCUS request or to create a new one. A user who chooses to execute an existing request will be shown an automatically generated list of FOCEXECs from which to pick. A user who chooses to create a new request will be placed in either TableTalk or PlotTalk, depending on whether reporting or creating a graph was chosen in the first step.

While the report or graph is being generated, a corresponding message will be displayed on the terminal screen. And, after the output is displayed, the user can choose to generate another report or graph, or else to exit.

The following figure illustrates the logic of the application FOCEXEC.

```
-START
-WINDOW SAMPLE MAIN
-*
-*Control is transferred from the above command
-*to window MAIN in window file SAMPLE.
-*
-IF &MAIN ...
-*
-*Control returns to the above command
-*from option "Exit?" in window MAIN,
-*from option "New Request?" in window EXECTYPE,
-*and from every selection in window EXECNAME.
-*
.
.
.
-GOTO START
-EXIT
```

Window	If option selected is...	Then go to:
MAIN	Report? Graph? Exit?	window EXECTYPE window EXECTYPE back to FOCEXEC
EXECTYPE	Existing Request? New Request?	window EXECNAME back to FOCEXEC
EXECNAME	The options in this window are a list of report and graph requests from which the user can select.	Control is transferred back to the FOCEXEC.

Creating the Application FOCEXEC

A FOCEXEC called SAMPLE will drive this application.

Begin by using the TED editor to create the FOCEXEC file SAMPLE. At the FOCUS prompt, type

```
TED SAMPLE
```

and press Enter. (In CMS, TED assigns the file type FOCEXEC unless you specify another file type. In MVS, you must specify ddname as follows:

```
FOCEXEC (SAMPLE)
```

Type in the following FOCEXEC. Note that the numbers on the left refer to explanatory notes. Do not type them in your FOCEXEC file, but read the notes as you go along. All commands that begin with a hyphen, such as -WINDOW, are Dialogue Manager commands, and they must begin in the first column. Dialogue Manager is discussed in Chapter 3, *Managing Applications With Dialogue Manager*.

You will notice that this application determines variable values in two ways: there are variables for which values are collected by windows, and variables which are set within the FOCEXEC using the -SET command.

```
-START
1.  -WINDOW SAMPLE MAIN
2.  -IF &MAIN EQ XXIT GOTO EXIT;
    -IF &MAIN EQ RPT GOTO GENERATE;
    -IF &MAIN EQ GRPH GOTO GENERATE;
    -GOTO START
    -***** GENERATE *****
3.  -GENERATE
4.  -IF &EXECTYPE EQ EXIST GOTO RPTEX ELSE GOTO NEWRPT;
5.  -RPTEX
6.  EX &EXECNAME
7.  -SET &FORMAT=IF &MAIN EQ RPT THEN REPORT
    -ELSE IF &MAIN EQ GRPH THEN GRAPH;
8.  -TYPE GENERATING &FORMAT
9.  -RUN
10. -GOTO START
11. -NEWRPT
12. -SET &PROCNAME=IF &MAIN EQ RPT THEN TABLETALK
    -ELSE IF &MAIN EQ GRPH THEN PLOTTALK;
13. &PROCNAME
14. -RUN
15. -GOTO START
    -***** EXIT *****
16. -EXIT
```

1. The -WINDOW command transfers control to the Window facility. SAMPLE is the name of the window file this application will use. (We will create it in this tutorial.) MAIN is the window where the procedure will begin.

Control will not return to the next line of the FOCEXEC until a window is processed for which no goto value has been assigned, in this case, EXECTYPE or EXECNAME.

2. The return value collected for &MAIN—collected from the window MAIN—is tested. The FOCEXEC branches to a label depending on its value.

If the return value for &MAIN is RPT or GRPH, the FOCEXEC will branch to -GENERATE; if XXIT, to -EXIT. Each return value corresponds to a selection on the menu window MAIN.

3. This label begins the GENERATE section of the FOCEXEC.
4. The value collected for &EXECTYPE (from window EXECTYPE) is tested and the FOCEXEC branches accordingly. Note that this value was collected from the window EXECTYPE while the Window facility was in control, without a prompt from Dialogue Manager.
5. This label begins the RPTEX section of the FOCEXEC.
6. The FOCUS command that will execute an existing report is stacked. The value of &EXECNAME—the name of the existing report—was collected while the window file was in control. The single quotation marks around &EXECNAME tell FOCUS to treat the value—which may contain more than one word (in CMS, for example, a file name and a file type)—as part of a single file identification.
7. The value of the variable &FORMAT is set according to the return value from the MAIN window. If the value was RPT, &FORMAT is set to REPORT; if the value is GRPH, &FORMAT is set to GRAPH.
8. A message containing the value of &FORMAT is displayed for the user while the stacked FOCUS request is executing.
9. -RUN executes the stacked command(s).
10. When the request output has been displayed, the FOCEXEC branches back to -START, where the user can choose to exit or to create another report or graph. All amper variable values collected in the previous round are cleared when the -WINDOW command is encountered.
11. This label begins the section NEWRPT.
12. This command sets the value of &PROCNAME to TABLETALK if the value of &MAIN is RPT, to PLOTTALK if the value is GRPH.
13. This line stacks the command TABLETALK or PLOTTALK.

14. -RUN executes the stacked command.
15. This command returns to -START, as in note 10.
16. This command ends FOCEXEC execution.

Creating the Window File

The -WINDOW command SAMPLE FOCEXEC tells FOCUS to look for a window file named SAMPLE and a window named MAIN. The complete list of windows used in this application is:

BORDER	A text display window used as a background display for the other windows.
BANNER	A text display window that introduces the application.
MAIN	A vertical menu from which the user can choose to create a graph or a report, or exit the application.
EXECTYPE	A vertical menu from which the user chooses to execute an existing procedure or create a new one.
EXECNAME	A file names window displaying all FOCEXEC files, from which the user can select one to execute. This window is seen only if the user opts to execute an existing report in EXECTYPE.

All these windows will be included in the window file named SAMPLE. You are going to start by building that window file.

- In CMS, when you use Window Painter to create a window file, the file is automatically created by the system on your A disk.
- In MVS, before you can use Window Painter to create a window file, a PDS must be allocated with ddname FMU, LRECL 4096, and RECFM F. BLKSIZE 4096 is recommended.

You can reach the FOCUS Window Painter Entry Menu by typing

`WINDOW [PAINT]`

at the FOCUS prompt, and pressing Enter.

The Entry Menu is the first screen you see:

```

+-----+
| Reporting      Ad hoc          Maintenance      Quit      |
+-----+
                                     +-----+
                                     | Add new information |
                                     | Update existing info |
                                     | Review Entries      |
                                     +-----+

```

Since you are creating a new window file, choose **NEW FILE**, and press Enter. The next screen you see prompts you to name the window file.

Since the FOCEXEC will look for a window file named SAMPLE, type

SAMPLE

and press Enter.

```

+-----+
| INSTRUCTIONS :   Move cursor to selection and hit ENTER |
|                  Use PF3 or PF12 to undo a selection    |
|                  Use PF1 for help                        |
+-----+

+-----+
|Select the window type: |
+-----+
|Menu (vertical)         |
|Menu (horizontal)       |
|Text input              |
|Text display            |
|File names              |
|Field names             |
|File contents           |
|Return value display    |
|Execution window        |
|Multi-Input window      |
+-----+

```

You will see a screen asking for a description of the window file.

Type

`Sample file for Window Painter tutorial`

and press Enter.

F O C U S W I N D O W P A I N T E R	
! INSTRUCTIONS :	Move cursor to selection and hit ENTER
!	Use PF3 or PF12 to undo a selection
!	Use PF1 for help
+-----+	
!Enter a description:	!
!Sample file for Window Painter tutorial.	!
+-----+	

Creating the Text Display Window Named BORDER

Now you are ready to create the first window. The screen that appears on your display is the Window Painter Main Menu. Select

`Create a new window`

and press Enter.

F O C U S W I N D O W P A I N T E R	
! INSTRUCTIONS :	Move cursor to selection and hit ENTER
!	Use PF3 or PF12 to undo a selection
!	Use PF1 for help
+-----+	
!Select one of the following: !	!
!Create a new window	!
!Edit an existing window	!
!Delete an existing window	!
!Run the window file	!
!Switch window files	!
!Utilities	!
!End	!
!Quit without saving changes	!
+-----+	

The Window Creation Menu asks what kind of window you want to create.

+-----+ INSTRUCTIONS : Move cursor to selection and hit ENTER Use PF3 or PF12 to undo a selection Use PF1 for help +-----+	
+-----+ Select the window type: +-----+	
Menu (vertical) Menu (horizontal) Text input Text display File names Field names File contents Return value display Execution window Multi-Input window +-----+	

The BORDER window is the first window you will create for the application. BORDER will supply a background border for other windows. It is a text display window, so select

`Text display`

and press Enter.

Next, you are asked to name the window. Type

`BORDER`

and press Enter.

File: SAMPLE F O C U S W I N D O W P A I N T E R	
+-----+ INSTRUCTIONS : Move cursor to selection and hit ENTER Use PF3 or PF12 to undo a selection Use PF1 for help +-----+	
+-----+ Enter a name for the window: +-----+	
BORDER +-----+	

The Window Description Screen appears next. This description does not appear when the window is displayed, but becomes part of the document file that Window Painter creates describing all windows in the file. Since the document file is very useful when writing your FOCEXEC, it is a good idea to enter a functional description here. To describe this window, type

This window borders all my screens.

and press Enter. The ability to annotate screens in this manner is very useful when selecting windows to edit.

```

File: SAMPLE          F O C U S    W I N D O W    P A I N T E R
+-----+
| INSTRUCTIONS :  Move cursor to selection and hit ENTER |
|               Use PF3 or PF12 to undo a selection    |
|               Use PF1 for help                        |
+-----+
+-----+
|Enter a window description:                             |
+-----+
|This window borders all my screens.                     |
+-----+

```

The Window Heading Screen comes next. Since you do not want a heading displayed on this window, simply press Enter to bypass it.

The Window Design Screen displayed now is nearly blank, with a cursor for you to position where you want the upper left-hand corner of BORDER to be. Leave the cursor where it is and press Enter.

A small box appears around the cursor: this is the window. You will now make the window larger. Using the arrow keys, move the cursor to the right edge of the screen, on the line just above the status line: this will be the new lower right corner of the window. Now press PF4 to resize the window. (PF4 functions as the SIZE key in the Window Design Screen.) The window has been resized so that its lower right corner is where you positioned the cursor: the window now fills the entire screen.

When resizing a window, remember that the window's lower right corner refers to the lower right corner of the window border, which is shown as a plus sign (+) on the screen. It is this corner that you are moving when you resize the window. On the other hand, the last row of the window refers to the last row that can contain data or text: this is the row immediately above the bottom border.

This window's border will form the background border for the other windows in this application.

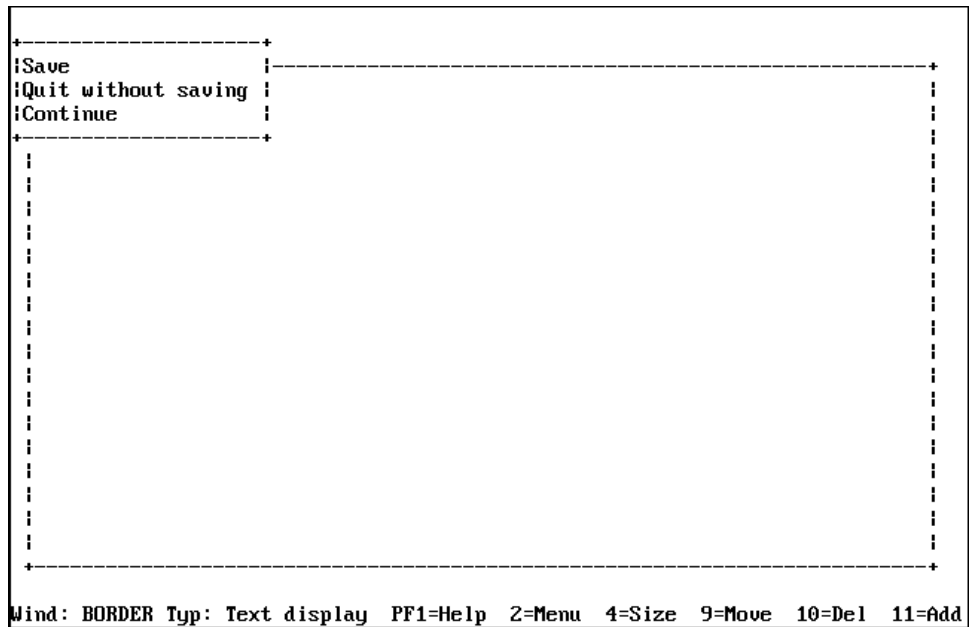
If you need help using the keyboard while in the Window Design Screen, press PF1 (the Window Painter Help key) to see the following display:

```
File: SAMPLE          F O C U S    W I N D O W    P A I N T E R

+-----+
|
|           Help: Text display and Return value display windows
|
| Use the arrow keys to move the cursor around on the screen.
| To enter text for a line, simply type that text in the window,
| for text display.
|
| PF01/PF13 : Help.
| PF02/PF14 : Main options menu.
| PF03/PF15 : Quit the Menu Design Screen.
| PF04/PF16 : Resize the window.
|
|           If you find that you do not have enough room in the window to
|           type the text you want, move the cursor to where you want the
|           new lower-right-hand corner to be, and press PF04 or PF16.
| PF05/PF17 : Set a window to go to if the current line is selected.
| PF06/PF18 : Set a return value for the current line.
| PF09/PF21 : Move the window.
|
|           To move the window, place the cursor where you want the new
|           upper-left-hand corner to be, and press PF09 or PF21.
| PF10/PF22 : Delete the line that the cursor is on.
| PF11/PF23 : Insert a line at the cursor position.
|
+-----+
```

Press Enter to continue.

Now that the window is complete, you should save it. Press PF3.



Press Enter to select Save. You will be returned to the Main Menu.

Creating the Text Display Window Named BANNER

BANNER is also a text display window, but is smaller than BORDER and contains text that identifies this application.

From the Window Painter Main Menu, select

Create a new window

and press Enter. Select

Text Display

and press Enter. The name of this window is

BANNER

and its description is:

Banner for application MAIN menu.

Enter this name and description just as you did for the BORDER window. When prompted for a heading, press Enter.

At the Window Design Screen, use the arrow keys to move the cursor two spaces to the right, and press Enter. Now position the cursor 64 more spaces to the right and two rows down, and press PF4 to resize the window.

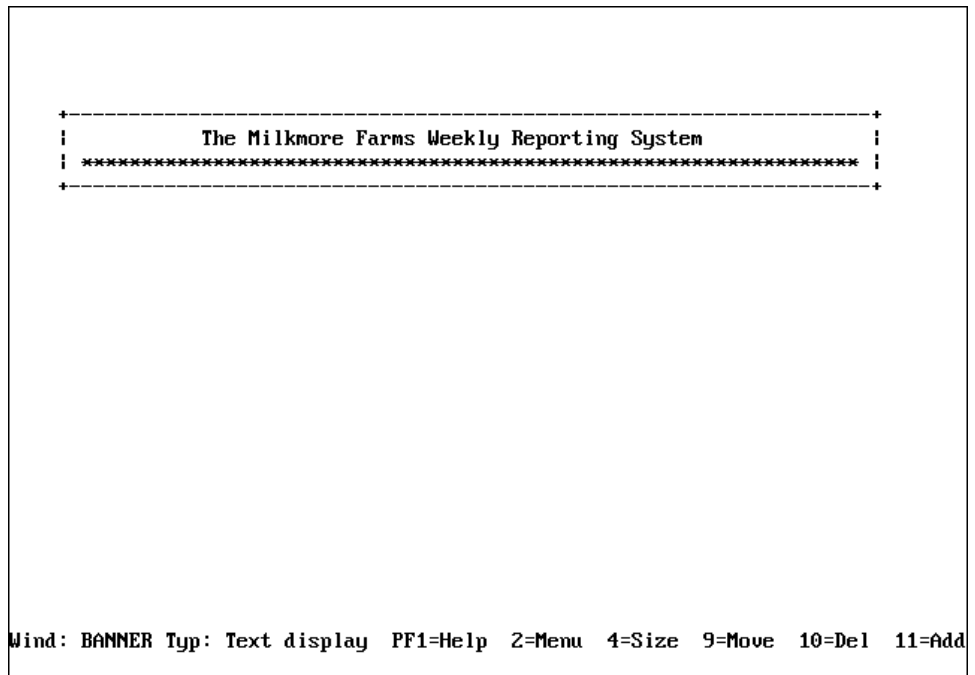
You will now enter text to be displayed in the window. Reposition the cursor on the first line within the window, ten spaces to the right of the window's left border, and type:

The Milkmore Farms Weekly Reporting System

Type a line of asterisks (*) all the way across the window's second line. (Begin at the second column within the window, because the first column of every window is protected.)

You will now center the banner in the width of the screen. Estimate where the upper left corner of the window would be if the window were centered. Position the cursor there, and then press PF9. The window moves to its new location. Repeat the process if you need to center it more precisely.

The window should look like this:



Press PF3 and save the window.

Creating the Vertical Menu Window Named MAIN

Now you will create the MAIN vertical menu window, which collects the amper variable &MAIN. Select

`Create a new window`

and press Enter.

BORDER and BANNER are text display windows, from which no options may be selected. Since MAIN, however, is a menu from which a selection must be made, choose

`Menu (vertical)`

and press Enter. Name the window:

`MAIN`

On the Description screen, type

`User can report, graph, or exit.`

and press Enter.

When prompted for a heading, type ten spaces, then

`Would you like to:`

and press Enter.

On the Window Design Screen, move the cursor five rows from the top and 20 columns from the left, and press Enter. The window will be created wide enough to contain the heading. Now position the cursor six rows below the window's bottom edge, and ten columns to the right of its right edge. Press PF4 and the window will be resized.

Type the following menu options as they appear below:

<table><tr><td>+</td><td>-----</td><td>+</td></tr><tr><td> </td><td>Would you like to:</td><td> </td></tr><tr><td>+</td><td>-----</td><td>+</td></tr><tr><td> </td><td>Create a report?</td><td> </td></tr><tr><td> </td><td>Create a graph?</td><td> </td></tr><tr><td> </td><td>Exit?</td><td> </td></tr><tr><td> </td><td></td><td> </td></tr><tr><td>+</td><td>-----</td><td>+</td></tr></table>	+	-----	+		Would you like to:		+	-----	+		Create a report?			Create a graph?			Exit?					+	-----	+
+	-----	+																						
	Would you like to:																							
+	-----	+																						
	Create a report?																							
	Create a graph?																							
	Exit?																							
+	-----	+																						
Wind: MAIN Type: Menu (vert) PF1=Help 2=Menu 4=Size 9=Move 10=Del 11=Add																								

Now you will assign goto and return values for each menu option. To assign either value to an option, the cursor must first be on that option.

Move your cursor back to

Create a report?

and press PF2 to display the pop-up Window Options Menu.

```

+-----+
|Exit this menu      |
|Goto value    PF5  |
|Return value  PF6  |
|FOCEXEC name      |
|Heading         |
|Description      | +-----+
|Show a window    | |         Would you like to:         |
|Unshow a window  | +-----+
|Display list     | | Create a report?                   |
|Hide list        | |                                     |
|Popup           (Off)| | Create a graph?                 |
|Help window      | |                                     |
|Line break       | | Exit?                               |
|Multi select (Off)| |                                     |
|Quit            PF3 | +-----+
|ACE security rule |
|Switch window    |
+-----+

Wind: MAIN      Typ: Menu (vert)  PF1=Help 2=Menu 4=Size 9=Move 10=Del 11=Add

```

Assigning a goto value tells the Window facility to display another window when this item is selected during execution.

In the next window of this application, the user will be prompted to either execute an existing report or create a new one. The window that displays that prompt will be called EXECTYPE, so the goto value of the first two menu options will be EXECTYPE.

Move the cursor to

Goto value

and press Enter.

In the space provided, type

EXECTYPE

and press Enter.

+-----+ !Enter name of next window to go to.! -----+ !Just 'Enter' for exit. ! you like to: ! +-----+ +-----+ !EXECTYPE ! ! Create a report? ! +-----+ ! ! ! Create a graph? ! ! ! ! Exit? ! ! ! +-----+ !	
Wind: MAIN Typ: Menu (vert) PF1=Help 2=Menu 4=Size 9=Move 10=Del 11=Add	

The return value collected by this window—&MAIN—will be tested in the FOCEXEC:

```
-START  
-WINDOW SAMPLE MAIN  
-IF &MAIN EQ XXIT       GOTOEXIT;  
-IF &MAIN EQ RPT        GOTO GENERATE;  
-IF &MAIN EQ GRPH       GOTO GENERATE;  
.  
.  
.
```

Now move the cursor to

Return value

and press Enter.

Type the value

RPT

as shown, and press Enter.

```

+-----+-----+
|Enter return value for the line:| Id you like to: |
+-----+-----+
|RPT          | reate a report? |
+-----+-----+
|              | Create a graph? |
|              | |
|              | Exit? |
|              | |
+-----+-----+

Wind: MAIN      Typ: Menu (vert)  PF1=Help 2=Menu 4=Size 9=Move 10=Del 11=Add

```

Exit the Window Options Menu by moving the cursor to

Exit this menu

and pressing Enter.

Now you will set the values for:

Create a graph?

Move the cursor to the second menu item, and press PF2.

Repeat the steps you just performed, assigning the goto value

EXECTYPE

and the return value:

GRPH

Leave the Window Options menu and move the cursor to

EXIT?

For this option, you will not assign a goto value. Since it exits to the FOCEXEC, there is no next window to be displayed.

Repeat the steps to assign the return value:

`XXIT`

With the Window Options Menu still on the screen, move the cursor to

`Display list`

and press Enter.

The display list may specify up to 16 windows to be displayed when this window is visible during execution. Since you want BORDER and BANNER to be displayed with MAIN, you must add them to the list.

The screenshot shows a main menu with the title 'Display list:' and a sub-menu titled 'Select one of these options:'. The sub-menu contains three options: 'Add to the list', 'Delete from the list', and 'Quit'. Below the sub-menu, there is a section titled 'Would you like to:' with three options: 'Create a report?', 'Create a graph?', and 'Exit?'. The menu is displayed in a simple, text-based format with dashed lines for borders and asterisks for alignment.

```
+-----+ +-----+
!Display list:! !Select one of these options:!
+-----+ +-----+
!Add to the list !
!Delete from the list!
!Quit !
+-----+ +-----+
! Would you like to: !
+-----+ +-----+
! Create a report? !
! Create a graph? !
! Exit? !
+-----+ +-----+
```

Select:

`Add to the list`

A list of windows appears, from which you select by moving the cursor and pressing Enter. The windows must be selected in the order in which they should appear, because they will be overlaid one on top of another when displayed. Select BORDER and BANNER for MAIN's display list, being certain to select BORDER first so that it will be displayed behind BANNER.

When you have finished, choose Quit to return to the Window Options Menu.

Quit the Window Options Menu and press PF3 to save MAIN.

Before moving on, look at what you have done so far. Select

`Run the window file`

and press Enter.

Select

MAIN

as the starting screen. Press Enter, and you will see a screen like this:

```
+-----+
|               |
|   The Milkmore Farms Weekly Reporting System   |
|*****|
|               |
+-----+

      +-----+
      |               |
      | Would you like to: |
      |_____|
      | Create a report? |
      | Create a graph? |
      | Exit?           |
      |               |
      +-----+
```

Position the cursor on the “Create a report” line. When you press Enter to continue the display, you will see an error message because EXECTYPE—the goto value—has not been created yet. Ignore it, and press Enter to continue. You will see a screen displaying amper variables for this window and their values. Press Enter to return to the Main Menu.

Creating the Vertical Menu Window Named EXECTYPE

So far you have created two text display windows and a vertical menu. The next window we will create will also be a vertical menu.

Select

Create a new window

from the Main Menu, and choose

Menu (vertical)

from the Window Creation Menu. Enter

EXECTYPE

as the window name.

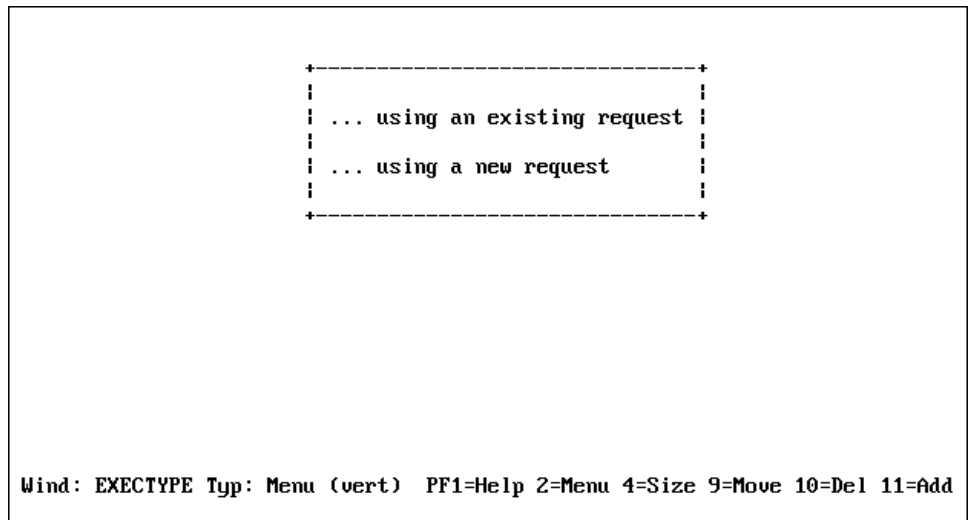
When prompted for a description, type

Create a new FOCEXEC or run existing one

and press Enter. When prompted for a heading, press Enter.

When the Window Design Screen appears, move the cursor 12 rows down the screen and 22 columns to the right, and press Enter. Now reposition the cursor four rows beneath the bottom edge of the window and 32 columns to the right of the right edge of the window, and press PF4 to resize it.

Type the following two menu options as they appear below:



The screenshot shows a window design screen with a dashed rectangular menu box. Inside the box, there are two menu options, each preceded by three vertical bars (| | |). The first option is "... using an existing request" and the second is "... using a new request". At the bottom of the screen, there is a status bar with the following text: "Wind: EXECTYPE Typ: Menu (vert) PF1=Help 2=Menu 4=Size 9=Move 10=Del 11=Add".

When you created the MAIN window, you used the Window Options Menu to set each return value and goto value. There is an easier way to set return and goto values using the PF6 and PF5 keys.

Pressing PF5 prompts you successively for a GOTO value, a Return value and a FOCEXEC name. When prompted for a GOTO value press Enter again and you will be prompted for the Return value. Enter EXIST and press PF5 again and you are prompted for FOCEXEC name. Just press Enter.

If you select

... using an existing request.

from the EXECTYPE menu, the file names window EXECNAME will be displayed next. EXECNAME will contain a list of existing FOCEXEC files from which you may choose.

Move the cursor to the second menu item.

Now you need to consider the return and goto values for this option.

If you choose to create a new report or graph request, EXECNAME will not be displayed. Rather, control must pass back to the FOCEXEC, which will execute these lines:

```
.  
.   
.   
-IF &EXECTYPE EQ EXIST GOTO RPTEX ELSE GOTO NEWRPT;  
.   
.   
-NEWRPT  
-SET &PROCNAME=IF &MAIN EQ RPT THEN TABLETALK  
ELSE IF &MAIN EQ GRPH THEN PLOTTALK;  
&PROCNAME  
-RUN
```

Since you want control to pass to the FOCEXEC if this option is chosen, you will not assign a goto value to it. Remember that during execution control passes to the FOCEXEC when an option without a goto value is selected.

The return value may be anything other than EXIST. For now, press PF6, and enter

```
NEXIST
```

Rather than create display and hide lists for EXECTYPE, make it a pop-up window. A pop-up window is displayed like any other window, but disappears when the user presses Enter. EXECTYPE pops up in front of MAIN.

Press PF2 to display the Window Options Menu, move the cursor to

```
Popup(Off)
```

and press Enter. You will see that (Off) changes to (On).

Exit the Window Options Menu, press PF3, and save the window.

Creating the File Names Window Named EXECNAME

Your final window is the file names window that displays a list of existing FOCUS report requests. On the Window Creation Menu, select:

```
File names
```

Name the window

```
EXECNAME
```

and type in the description:

```
Select an existing FOCEXEC from list.
```

Enter an explanatory heading:

```
Select the request you want to execute and press ENTER:
```

You will be prompted for file-identification criteria. Type

* FOCEXEC

and press Enter.

File: SAMPLE	FOCUS	WINDOW	PAINTER
+-----+			
	INSTRUCTIONS :		Move cursor to selection and hit ENTER
			Use PF3 or PF12 to undo a selection
			Use PF1 for help
+-----+			
+-----+			
Enter the file name criteria (e.g. * MASTER)			
or & variable name containing the criteria:			
+-----+			
* FOCEXEC			
+-----+			

- In CMS, when the application is executed, this will select all files having the file type FOCEXEC.
- In MVS, when the application is executed, this will select all members of ddname FOCEXEC.

On the Window Design Screen, move the cursor two rows down and press Enter. Use PF9 to center the window on the screen. Resize the window: reposition the cursor two columns to the right of the window's right edge and ten rows below the window's bottom edge, and press PF4.

Since only BORDER should be displayed with this window, add BANNER, MAIN, and EXECTYPE to the hide list and add BORDER to the display list.

When the user selects a file name from this window during execution, that file name will automatically be collected as the return value. You cannot set the return value any other way for this type of window.

In the FOCEXEC, that return value will be plugged into the line

EX &EXECNAME

and the report or graph request will be executed.

But in order for this to happen, you must return control to the FOCEXEC. Therefore, you will assign no goto value to this window.

If you want to change the file identification criteria of a file names window (or of a field names or file contents window) after it has been created, change the “return value.” Although these two window types cannot have their actual return values set when the window is created or edited, the “return value” which is displayed and can be set is actually the window’s file identification criteria. You can change the file identification criteria just as you would change the actual return value of a vertical menu window.

Exit from the Window Options Menu, press PF3, and save the window. The window file is complete. Exit from Window Painter.

Executing the Application

To execute the SAMPLE FOCEXEC, at the FOCUS prompt, type

`EX SAMPLE`

and press Enter. When prompted to choose a new or existing FOCEXEC, select

`... using a new request.`

unless you have created one in an earlier FOCUS session. The application will execute PlotTalk or TableTalk. If you save the request you create, you can try the SAMPLE FOCEXEC again, and execute the new request by selecting:

`... using an existing request.`

This completes the tutorial.

Window Painter Screens

The creation of windows is itself an automated window-driven process. There are six major screens:

- The Entry Menu
- The Main Menu
- The Window Creation Menu
- The Window Design Screen
- The Window Options Menu
- The Utilities Menu

These screens assist you whenever you create or edit windows.

Invoking Window Painter

To invoke Window Painter, type the WINDOW PAINT command at the FOCUS prompt and press Enter.

Syntax

How to Invoke Window Painter

```
WINDOW [PAINT [filename]]
```

where:

PAINT

Is optional.

filename

Is the name of the window file that you want to work with.

In CMS, this is a file name. The file must have a file type of FMU.

In MVS, this is a member name. The member must belong to ddname FMU.

If you do not specify file name, you will begin your Window Painter session at the Entry Menu, where you can choose a window file to use or can create a new window file. If you do specify file name, you will skip the Entry Menu and begin your Window Painter session at the Main Menu, working with the window file you specified.

If the file name does not exist, you will be asked if you want to create a new file. If not, the Window Painter Entry Menu will be displayed.

Entry Menu

You can reach the Window Painter Entry Menu by typing

`WINDOW [PAINT]`

at the FOCUS prompt, and then pressing Enter.

The Entry Menu is the first screen you see:

```

File: SAMPLE          F O C U S   W I N D O W   P A I N T E R

+-----+
| INSTRUCTIONS :   Move cursor to selection and hit ENTER |
|                 Use PF3 or PF12 to undo a selection    |
|                 Use PF1 for help                       |
+-----+

+-----+
|Select the window file:                                |
+-----+
|New File   Create a new file                            |
|TEST      This is a test.                               |
|SAMPLE    Sample file for Window Painter tutorial.     |
+-----+

```

The Entry Menu invites you to choose a window file in which to work. If you are creating windows for a new application, you should start a new window file. If you are maintaining or creating windows for an existing application, use the window file that corresponds to your application.

When you become comfortable working with windows, you can write FOCEXECs that include branching between window files. Refer to *Transferring Control in Window Applications* on page 8-22 for a detailed discussion on branching and transferring control.

Main Menu

Once you have selected a window file from the Entry Menu, or entered the WINDOW PAINT command with the file name option, the Main Menu appears:

F O C U S W I N D O W P A I N T E R	
+-----+	
INSTRUCTIONS :	Move cursor to selection and hit ENTER
	Use PF3 or PF12 to undo a selection
	Use PF1 for help
+-----+	
+-----+	
	Select one of the following:
+-----+	
	Create a new window
	Edit an existing window
	Delete an existing window
	Run the window file
	Switch window files
	Utilities
	End
	Quit without saving changes
+-----+	

The following table summarizes the options on the Main Menu, along with illustrations of screens that appear when you select some of the options:

Menu Option	Description
Create a new window	Brings up the Window Creation Menu. You can select the type of window you want to create.
Edit an existing window	Brings up a list of windows in your current window file. You can select the one you want to edit.

File: SAMPLE	FOCUS WINDOW PAINTER
+-----+	
INSTRUCTIONS :	Move cursor to selection and hit ENTER
	Use PF3 or PF12 to undo a selection
	Use PF1 for help
+-----+	
+-----+	
	Select window to edit:
+-----+	
BORDER	This window borders all my screens.
BANNER	Banner for application MAIN menu.
MAIN	User can report, graph, or exit.
EXECTYPE	Create a FOCEXEC or run an existing one.
EXECNAME	Select an existing FOCEXEC from list.
+-----+	

Menu Option	Description
Delete an existing window	Brings up a list of windows in your current window file. You can select the one you want to delete.

File: SAMPLE	FOCUS WINDOW PAINTER
+-----+	
INSTRUCTIONS :	Move cursor to selection and hit ENTER
	Use PF3 or PF12 to undo a selection
	Use PF1 for help
+-----+	
+-----+	
	Select window to delete:
+-----+	
BORDER	
BANNER	
MAIN	
EXECTYPE	
EXECNAME	
+-----+	

Menu Option	Description
Run the window file	<p>Brings up a list of windows in your current window file. You can select the one from which you want to start running the window file.</p> <p>After the window file is run, the windows' amper variable values are displayed. The display includes the first 20 characters of each value.</p> <p>This option shows you how your windows work without executing the FOCEXEC. Use this option to test your window file.</p>
Switch Window files	Returns you to the Window Painter Entry Menu, from which you can select another window file. The previous window file is saved whenever you switch window files.
Utilities	Brings up the Utilities Menu, which is discussed in <i>Utilities Menu</i> on page 8-72.
End	Returns you to native FOCUS. All work that you saved during the Window Painter session is kept.
Quit without saving	Returns you to native FOCUS. All work that you saved during the Window Painter session is discarded.

Window Creation Menu

You can reach the Window Creation Menu by selecting

`Create a New Window`

from the Main Menu. You will see the following screen:

```

+-----+
| INSTRUCTIONS : Move cursor to selection and hit ENTER |
|               Use PF3 or PF12 to undo a selection   |
|               Use PF1 for help                       |
+-----+
|
|               +-----+
|               |Select the window type: |
|               +-----+
|               |Menu (vertical)         |
|               |Menu (horizontal)       |
|               |Text input              |
|               |Text display            |
|               |File names              |
|               |Field names             |
|               |File contents           |
|               |Return value display    |
|               |Execution window        |
|               |Multi-Input window     |
|               +-----+
|

```

You will first need to select the type of window you will create. You will then be asked to enter an 8-character name and an optional 40-character description. These are for your use only; they do not appear in the window during execution.

For a vertical menu, horizontal menu, text input, text display, file names, field names, file contents, multi-input, or return value display window, you are prompted to supply a 60-character heading.

For a text input window, you are prompted to choose the format of the text entry field (alphanumeric, with all text translated to uppercase; alphanumeric, with no case translation; or numeric). Later, in the Window Design Screen, you can make the length of the text entry field shorter than the window's header length by typing a single character in the window immediately following the last desired field position, or by typing characters continuously from the first field position to the last desired field position.

For a file names, field names, or file contents window, you are prompted to produce file-identification criteria that can consist of an amper variable, a complete file identification, or (for file names windows) a file specification which includes an asterisk (for example, * MASTER).

The asterisk is used as a wildcard character: it indicates that any character or sequence of characters can occupy that position. In CMS, an asterisk used in file-identification criteria can be embedded (for example, *DEPT FOCEXEC); the asterisk can be used in the file name, the file type, and the file mode. In MVS, the asterisk can be used as the member name but not in the ddname.

If an amper variable is used, you can prompt for the file identification criteria at run time.

- File-identification criteria in CMS must specify the file name first, the file type second, and an optional file mode third. If the file mode is not specified, it defaults to an asterisk.
- File-identification criteria in MVS must specify the member name first and the ddname second.

If you are creating a field names window, your file-identification criterion is the name of a Master File.

In addition, you can create execution windows containing FOCUS commands such as Dialogue Manager commands or TABLE requests. You will be prompted for the window name and heading. Once a window has been specified, you will see the Window Design screen.

For complete information about the types of windows you can create in Window Painter, see *Types of Windows You Can Create* on page 8-4.

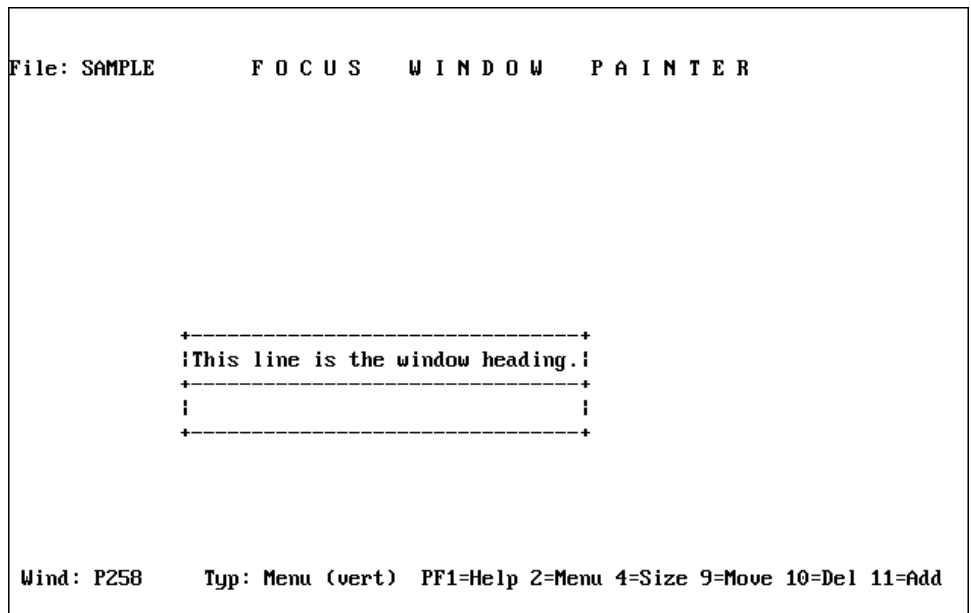
The next screen displayed is the Window Design Screen, discussed in *Window Design Screen* on page 8-59. This screen enables you to enter information, and position and size your window.

Window Design Screen

In this screen you design the appearance and functionality of your windows. It appears during the window creation process, when you press Enter after typing the heading of your window.

The Window Design Screen consists of a blank screen, a cursor, and text asking you to move the cursor to the starting position for the window. This starting position becomes the upper left corner of the window. Use the cursor arrow keys to move the cursor to the place where you want the upper left corner of the window to be, and press Enter.

When you press Enter this time, the window appears, with its heading at the top. You can enlarge it, type text in it, and move it around the screen.



The Window Design Screen lets you use the keyboard to manipulate the window you are creating.

The following chart summarizes Window Design Screen key functions in all window types.

PF Key	Function
PF1	Displays a window of help information.
PF2	Displays the Window Options menu. This menu is discussed in <i>Window Options Menu</i> on page 8-61.
PF3	Displays the exit menu. You can select: <ul style="list-style-type: none"> • Exiting from the Window Design Screen while saving your work. • Quitting from the Screen without saving your work. • Continuing your work.
PF4	Resizes the window. First move the cursor to the desired position of the window's lower right corner. When you press PF4, the window's upper left corner remains in the same position; the window's lower right corner moves to the current cursor position. If the window size is reduced, nothing in the window is deleted; all window contents beyond the window border can be seen by scrolling the window.
PF5	Gets the GOTO value, the Return value and the FOCEXEC name for the active window.
PF6	Sets the return value of the line that the cursor is on.
PF7	Scrolls the window up if the window contents extend beyond the top border.
PF8	Scrolls the window down if the window contents extend beyond the bottom border.
PF9	Moves the window. First move the cursor to the desired position of the window's upper left corner. When you press PF9, the window's upper left corner (the + in the border) moves to the current cursor position. The rest of the window moves accordingly.
PF10	Deletes the line of window contents identified by the current cursor position. If the window contents do not extend beyond the window borders, then the window itself will be reduced by one line.
PF11	Adds one line of window contents beneath the line identified by the current cursor position. If the window contents do not extend beyond the window borders, then the window itself will increase by one line.
PF12	Provides the same function as the PF3 key.
PF13 - PF24	These keys provide the same functions as the corresponding keys PF1 - PF12.

If a window's contents extend beyond a top or bottom border, then the message

(MORE)

is displayed on that border. This reminds you that there are more lines of contents that are hidden beyond that border. You can view these lines by scrolling the window toward the border. When the window is used in an application, the user can also scroll the window to see all of the contents.

The display line at the bottom of the Window Design Screen shows instructions or information. When you first see the Window Design Screen, the display line tells you to move the cursor and press Enter. When you press Enter, the display line shows the name of the window file, and the name and type of window being created; it also tells which keys to press for the HELP function, the SIZE function, and the Window Options Menu.

Window Options Menu

When the Window Design Screen is displayed, pressing PF2 brings up the following Window Options Menu:

Exit this menu Goto value Return value FOCEXEC name Heading Description Show a window Unshow a window Display list Hide list Popup (Off) Help window Line break Multi select (Off) Quit PF3 Conceal option Switch window	<table border="1"> <tr> <th>Would you like to:</th> </tr> <tr> <td>Create a report?</td> </tr> <tr> <td>Create a graph?</td> </tr> <tr> <td>Exit?</td> </tr> </table>	Would you like to:	Create a report?	Create a graph?	Exit?
Would you like to:					
Create a report?					
Create a graph?					
Exit?					
Wind: MAIN Typ: Menu (vert) PF1=Help 2=Menu 4=Size 9=Move 10=Del 11=Add					

The following table summarizes the options on this menu, along with illustrations of screens that appear when you select some of the options:

Menu Option	Description
Goto value	<p>Selecting this option lets you specify the next window in the path from this selection field or window. You will be asked to supply the name of the window. (It does not matter whether or not this window exists. You can create it later, but remember the name you chose for it.)</p> <p>In menu windows, goto values are assigned to each menu item. In other windows, there is a single goto value for the entire window.</p> <p>To assign a goto value, your cursor must be on the proper line when the Window Options Menu is brought up. Select Goto value from the Window Options Menu and you will be prompted to enter the name of the window that is the target of the goto. Type the name in the space provided and press Enter again. The goto value is assigned.</p>

```
+-----+
|Enter name of next window to go to.| -----+
|Just 'Enter' for exit.              | you like to: |
+-----+ -----+
|EXECTYPE |      | Create a report?      |
+-----+      |      |                  |
|                  | Create a graph?      | |
|                  |      |                  |
|                  | Exit?                |
|                  |      |                  |
+-----+      +-----+
```

Wind: MAIN Typ: Menu (vert) PF1=Help 2=Menu 4=Size 9=Move 10=Del 11=Add

Menu Option	Description
Return value	<p>The return value supplies a value for an amper variable. If the user selects this field during execution, the return value you have assigned is plugged into the amper variable in your FOCEXEC. Return values are assigned to each menu item in menu windows, and one per window for other window types. The only exceptions are the multi-input window, whose return value is the name of the input field occupied by the cursor when you pressed Enter or a PF key, and the return value display window, which does not have a return value but instead displays other windows' return values. The return value for a Multi-Select window is the number of selections.</p> <p>To assign a return value, your cursor must be on the proper line when the Window Options Menu is brought up. Select Return value from the Window Options Menu and you will be prompted to enter a return value. Note that for file names, field names, and file contents windows, the value that you enter is the file-identification criterion for that window. Type the value in the space provided and press Enter again. The return value is assigned.</p>

```

+-----+-----+
|Enter return value for the line:| Id you like to: |
+-----+-----+
|RPT          | reate a report? |
+-----+-----+
|              | Create a graph? |
|              | |
|              | Exit? |
|              | |
+-----+-----+

```

Wind: MAIN Typ: Menu (vert) PF1=Help 2=Menu 4=Size 9=Move 10=Del 11=Add

Menu Option	Description
FOCEXEC name	Attaches a FOCEXEC to each menu selection of the window. The FOCEXEC is executed when the menu item is selected.
Heading	Changes the heading of any window you are working on. You can also add or remove a heading.
Description	Changes the description of any window you are working on.
Show a window	Used only during window editing, brings another window onto the screen for reference. You cannot edit the second window.
Unshow a window	Removes the shown window from the display.

Menu Option	Description
Display list	<p>Enables you to specify a list of up to 16 windows that will be visible when this window is displayed during execution.</p> <p>Note that if part of a window on the display list extends beyond the window border or does not fit on the screen, it cannot be scrolled.</p> <p>As many as 16 windows can be displayed on the screen at one time. This applies to all windows on the screen (that is, a window displayed during execution, windows displayed when executed previously and not hidden afterward, and windows displayed because specified on a display list). The window facility interprets each window heading as a separate window: if all of the windows have headings, 16 of them can be displayed on the screen at one time.</p>

```

+-----+
!Display list:!
+-----+
!BORDER  !
!BANNER  !
+-----+

+-----+
!Select one of these screens:!
+-----+
!EXECTYPE!
!EXECNAME!
+-----+

+-----+
!          Would you like to:          !
+-----+
! Create a report?                      !
!                                       !
! Create a graph?                      !
!                                       !
! Exit?                                !
!                                       !
+-----+

Wind: MAIN      Typ: Menu (vert)  PF1=Help 2=Menu 4=Size 9=Move 10=Del 11=Add

```

Menu Option	Description
Hide list	Allows you to specify windows that will not appear when this window is displayed during execution. You can specify up to 16 specific windows or all windows in the window file. If you select "All," all the windows will be hidden except those in the display list. -- If you do not hide a window that was displayed, it will remain on the screen until another window that includes it on a hide list is displayed during execution.

+-----+

|Hide list: |

+-----+

|EXECNAME |

+-----+

+-----+

|Select one of these options:|

+-----+

|All |

|BORDER |

|BANNER |

+-----+

+-----+

| Would you like to: |

+-----+

| Create a report? |

| Create a graph? |

| Exit? |

+-----+

Wind: MAIN Typ: Menu (vert) PF1=Help 2=Menu 4=Size 9=Move 10=Del 11=Add

Menu Option	Description
Popup (Off/On)	Makes the window disappear when the user presses Enter during execution. Defaults to OFF, which leaves the window on screen. Set Popup to OFF with text display windows as they do not work even if set to ON.
Help window	<p>Lets you display information about a window or a menu item when a user presses PF1 (the Window facility HELP key) during execution. The information displayed is text within a specified Help window.</p> <p>Note that if the PFKEY option is specified in the -WINDOW command, you will have to explicitly set a PF key as the HELP key, as described in <i>Testing Function Key Values</i> on page 8-26.</p> <p>When selecting the Help window option, you will be asked to supply the name of the Help window file that contains the Help window. Next, you will be asked to supply the name of the Help window itself. The Help window can be an existing window, or one that you will create.</p> <p>If the Help window displays field names, it qualifies duplicates with the segment name.</p> <p>You can use any window type for a Help window. A text display window is easiest, except when you want to supply different help information for each item in a vertical menu, horizontal menu (that is, item-specific help).</p> <p>If you wish to assign item-specific help, use a file contents window that displays a file containing text in the following format:</p> <pre>=>HELPPFILE => menu item this is the Help message you want the user to see.</pre> <p>where:</p> <pre>=></pre> <p>Is entered with an equal sign (=) and a greater-than sign (>).</p> <pre>HELPPFILE</pre> <p>Must be uppercase.</p>

<p>Help window (continued)</p>	<p><i>menu item</i></p> <p>Is the exact text of the menu item. Any blank spaces that precede this text in the menu must also precede this text here in the Help file. Note that at least one blank space always precedes the menu item text in a vertical menu, horizontal menu, or multi-input window.</p> <p>For example, if the first three lines of a vertical menu are</p> <pre>(1) Generate a sales report (2) Generate a stock report</pre> <p>and there are three blank spaces between the left border of the window and the beginning of the text, then the file containing help text could look like this:</p> <pre>=>HELPPFILE => (1) Generate a sales report This option displays a list of existing sales report requests, and lets you select one of these requests to execute. => (2) Generate a stock report This option displays a list of existing stock report requests, and lets you select one of these requests to execute.</pre> <p>The lines immediately following the menu item text are displayed when the user positions the cursor on the menu item and presses PF1.</p> <p>In some cases you may wish to assign topic-specific help, but you may want the help text for some of the topics to be contained in a separate file. In these cases, on the line following the menu item text, replace the help message with the file identification of the file containing that menu item's help message.</p> <p>In CMS, use this file-identification format:</p> <pre>FILENAME= filename filetype [filemode]</pre> <p>In MVS, use this file-identification format:</p> <pre>FILENAME= membername ddname</pre> <p>To assign one set of instructions that can be used for multiple menu items, use the following syntax:</p> <pre>=>DEFAULT This text appears when you have not written topic-specific help.</pre>
---	---

Help window <i>(continued)</i>	<p>The DEFAULT text must be the last section in the Help file.</p> <p>Lines beginning with an * are comment lines that are not displayed.</p> <p>What follows is an example of a topic-specific Help file for the Main Menu used in the tutorial.</p> <pre>=>HELPPFILE *Help file for tutorial/Main Menu => Create a report? Choose this option if you wish to create a new report. => Create a graph? Select this option if you wish to create pie charts, bar charts or other graphics. => Exit? If you wish to leave the application, choose this option.</pre>
Line-break	<p>Formats the contents of the return value display window. This option is set when designing the windows from which you collect the return value(s) to be displayed.</p> <p>When you select this option, you will see:</p> <pre>None New line before value New line after value Both</pre> <p>where:</p> <pre>None</pre> <p>Places return value directly after preceding value. If there is not enough room on this line, return value is placed on the next line.</p> <pre>New line before value</pre> <p>Places return value on the next line.</p> <pre>New line after value</pre> <p>Places return value on the same line as preceding value. Places next return value on next line.</p> <pre>Both</pre> <p>Places return value on a line by itself.</p>

<p>Multi-Select</p>	<p>Enables you to select multiple items from one window. The number of items you select is collected as the return value from that window; each selected item's return value is stored in a temporary file in memory. You can later retrieve these stored values for use in a FOCEXEC. Values for up to eight windows can be stored at one time.</p> <p>When you select this option, you will see:</p> <pre>-Select Multi(On)</pre> <p>During execution, the user selects individual values by pressing PF9. After all selections have been made, the user presses Enter.</p> <p>Note that when the -WINDOW command is issued with the PFKEY option, the PF9 key cannot be used to make selections unless a SET command is issued before the -WINDOW command. For example:</p> <pre>SET PF09=SELECT</pre> <p>You can also set a different PF key for selecting multiple items.</p> <p>A Multi-Select window can have no more than one goto value. Although in a vertical menu window you can assign a different goto value to each menu item, only the value assigned to the first item is effective.</p> <p>The return value collected for a window using the Multi-Select option is the number of values selected by the user.</p> <p>To retrieve the individual values, issue a special WINDOW call, as follows:</p> <pre>-WINDOW windowfile windowname GETHOLD</pre> <p>where:</p> <pre>windowfile</pre> <p>Is the name of the window file.</p> <pre>windowname</pre> <p>Is the name of the Multi-Select window.</p> <pre>GETHOLD</pre> <p>Is the special parameter that retrieves one value at a time from the temporary file.</p> <p>The value is assigned to the variable &windowname.</p>
----------------------------	--

Multi-Select (continued)	<p>The GETHOLD option requires at least two -WINDOW commands in your FOCEXEC. The first -WINDOW command (without the GETHOLD option) transfers control to the Window facility where a Multi-Select window is used. The second and subsequent -WINDOW commands use the GETHOLD option to retrieve the stored amper variables collected in a particular Multi-Select window.</p> <p>For each value to be retrieved, you will need a -WINDOW command with the GETHOLD option. Each value will be stored in &windowname. If you wish to use this value, we recommend assigning it to another variable. For example, if the return value has the value 4, you would issue the special -WINDOW command four times; each time you would collect the value from &windowname. Alternatively, you could perform a loop.</p> <p>Note that -WINDOW with the GETHOLD option will not transfer control from the FOCEXEC to the Window facility.</p>
Quit	Returns you to the Window Painter Entry Menu.
Input fields	Input fields pertain to Multi Input Windows. Selecting the field takes you to that field.
Menu text	Specifies a line of descriptive text, up to 60 characters long, for items on a horizontal menu. Use the Text line option to position the text.
Text line (x+1)	On a horizontal menu, positions descriptive text one or two lines above or below the menu. Valid values are x+1 or x+2 to place the text above the horizontal menu, x-1 or x-2 to place the text below the horizontal menu. Use the Menu text option to define the descriptive text.
Pulldown (off/on)	If the setting is ON, placing the cursor on an item in a horizontal menu can display an associated pulldown menu. The default setting is OFF. Turn the setting ON by positioning the cursor on this option and pressing Enter. — The pulldown menu must be a vertical menu and must be assigned as the goto value for the horizontal menu item. Note that setting Pulldown ON automatically shuts off Menu Text.
Switch window	Enables you to work on and move between two windows. When you select this option, you can create a new window, or edit an existing window without returning to the Main Menu.

Utilities Menu

If you select the Utilities option from the Window Painter Main Menu, the Utilities Menu will be displayed:

INSTRUCTIONS : Move cursor to selection and hit ENTER Use PF3 or PF12 to undo a selection Use PF1 for help
<div><div>Select one of the following:</div><div>Document the file Change the file description Compress the file Rename a window Copy a window Select the start window Create a transfer file Quit the Utilities Menu</div></div>
File: SAMPLE F O C U S W I N D O W P A I N T E R

The following table summarizes the options on this menu, along with illustrations of screens that appear when you select some of the options:

Menu Option	Description
Document the file	<p>When you select this utility, Window Painter creates documentation of the window file. You can display the document on the screen using TED or another system editor, or send it to a printer or disk file.</p> <p>In CMS, this option creates a file with file type TRF on your A disk.</p> <p>In MVS, this option creates a member of the TRF PDS; that PDS must have already been allocated. However, creating a PDS is not necessary if you are only going to use the documentation file during the current FOCUS session: Window Painter will temporarily allocate the PDS.</p> <p>This document contains detailed information about all the windows in the window file. It shows you the kinds of windows, their structure and format, and any options you have assigned from the Window Options Menu, including return and goto values. The text you enter when prompted for a window file description or individual window description is part of this document.</p> <p>The document is especially useful when creating a FOCEXEC, since it provides return and goto values in addition to other information.</p> <p>Note: If you create another file with the same name, the file is not overwritten. It is appended.</p>

```
* WINDOW FILE NAME=SAMPLE
* DESCRIPTION='Sample file for windows tutorial'
* WINDOW NAME=MAIN, TYPE=Menu (vertical)
* DESCRIPTION='User can report, graph, or exit.'
* ROW= 6,COLUMN=23,HEIGHT= 7,WIDTH=38,WINDOW= 7,POPUP= 0,BORDER= 2,HEADLEN=28,
* RETURN=None
* MULTI=Off
* HEADING:
* Would you like to:
* WINDOW DATA:          GOTOS:          VALUES:
* 1.' Create a report?    ',',EXECTYPE    ',',RPT
* 2.' Create a graph?    ',',EXECTYPE    ',',GRPH
* 3.' Exit?              ',',          ',',XXIT
* DISPLAY LIST:
* BORDER
```

Menu Option	Description
Change the file description	Changes the description of the current window.
Compress the file	This utility is provided to help you save space in memory. It allows space made available by deleted or edited windows to be reused.
Rename a window	When you select this utility, you see a list of the windows in the current window file. You can change the name of any of these windows.
Copy a window	<p>This function copies a window from one window file to another, or duplicates it within the same file.</p> <p>The copy function is useful when you create a new application, or need to add windows to an existing application, and want the windows to look like those you have already created. You can copy any window and edit it to conform to the new application.</p>
Select the start window	Enables you to choose a default start window. This window is the first to be entered if a specific window is not selected upon startup. If a default start window is not explicitly chosen, FOCUS will select the first window created to be the start window.

Menu Option	Description
Create a transfer file	<p>Creates a file to be transferred for use with the Window facility in another FOCUS environment.</p> <p>In CMS, this option creates a file with file type TRF on your A disk.</p> <p>In MVS, this option creates a member of the TRF PDS; that PDS must have already been allocated.</p>
Quit the utilities menu	Returns you to the Main Menu.

Transferring Window Files

If you use FOCUS in more than one operating environment, you can transfer an existing window file from one environment to be used in another environment. For example, if you have a fully-developed window application in PC/FOCUS, and you want to develop a similar application in mainframe FOCUS, you can transfer the PC/FOCUS window file to mainframe FOCUS; this saves you the trouble of recreating the window file from scratch in mainframe FOCUS.

You can transfer a window file to a new environment in four simple steps:

1. Create a transfer file from the original window file using Window Painter.
2. Transfer the new file to the new environment using FTP.
3. Edit the transferred file in TED, if necessary.
4. Compile the transferred file using the WINDOW COMPILE command.

These steps are described in the following topics.

Creating a Transfer File

The window files that you design in Window Painter are compiled files; before a window file can be transferred to another environment, a user-readable source code version must be created. This user-readable file is called a transfer file, and is created using the transfer file option of Window Painter.

- In CMS, this Window Painter option automatically creates a transfer file with a file type of TRF on your A disk.
- In MVS, this Window Painter option automatically creates a new member of the PDS allocated to ddname TRF; the PDS must already have been allocated (with LRECL between 80 and 132 and RECFM FB). However, it is not necessary to create the PDS if you are only going to use the transfer file during the current FOCUS session: Window Painter will temporarily allocate the PDS.
- For information about the transfer files created by FOCUS Window Painter in other operating environments, see the appropriate FOCUS Users Manual for those environments.

To convert a window file to a transfer file, go to the Window Painter Utilities Menu and select:

`Create a transfer file`

You will then be prompted for the name of the new transfer file. Enter any name that you wish; it can have the same name as the window file, or an entirely new name. In CMS the name that you enter is the file name; in MVS it is the member name.

Note that you should not give the transfer file a name already assigned to a window documentation file. Also, you should not give the transfer file a name already assigned to an existing transfer file unless you want to merge the two files, as described below. See the appropriate operating environment topic in the *Overview and Operating Environments* manual for more information about duplicate window transfer and window documentation file names.

You will be asked to select which window(s) you want to transfer. You can select

`All`

to transfer all of the windows in the current window file, or you can select any single window in the file. This is the last step in creating a transfer file.

Note that you can merge transfer files: if a transfer file already exists for your window file, and you only need to add a new window to it, you can give the new transfer file the same name as the old one, and then select the new window. Window Painter will merge the source code for the new window into the existing file, so that you have a single complete transfer file.

Transferring the File to the New Environment

Once the transfer file exists, it can be transferred to the new environment using FTP.

Editing the Transfer File

Window facility features introduced in one FOCUS release may not be fully supported in earlier releases. Because different operating environments may be running different releases of FOCUS, the transfer file created by the FOCUS Window facility in one environment may contain features not fully supported by the Window facility in another environment.

If your transfer file contains Window facility features not fully supported in the new environment, you may need to remove or fine-tune those features. If, on the other hand, the new environment supports features not supported in the original environment, you can add those features to the transfer file. Adding, removing, and fine-tuning features can be done by simply editing the transfer file.

The Format of the Transfer File

The transfer file is a user-readable source code listing of all of the windows, and their features, that were included from the original window file. You can remove or fine-tune an unsupported feature by simply editing or deleting the appropriate line in the transfer file. You can accomplish this by using TED or any other editor.

Each transfer file contains:

- One set of window file attributes describing the file.
- For each window defined in the file, one set of window attributes describing that window.
- For each line in each window, one set of attributes describing that line.

If any attribute is not specified in the transfer file, it defaults to a value of zero or blank (depending on whether the value is normally numeric or alphanumeric).

Attribute	Description
FILENAME	The name of the original window file.
DESCRIPTION	A comment field describing the file.
WINDOWNAME	The name of the window.
TYPE	The type of window: <ol style="list-style-type: none">1. Vertical menu2. Text input window3. Text display window4. Horizontal menu5. File names window6. Field names window7. File contents window8. Return value display window9. Execution window10. Multi-input window
COMMENT	A comment field describing the window.
TRANSLATE	Type of input for text input windows (Type 2). <ol style="list-style-type: none">0 Allow mixed case input.1 Allow numeric input only.2 Translate input to uppercase.
ROW	The row number of the upper left corner of the window.
COLUMN	The column number of the upper left corner of the window.
HEIGHT	The height of the window data (the number of lines of window data, not the height of the actual window frame). If there are more data lines than will fit in the window frame, the PF7 and PF8 keys can scroll the window.
TEXT LINE	Position of menu text. Values are: +1, +2, -1, -2.
WIDTH	The width of the window frame, not including the border.

Attribute	Description
INPUT FIELDS	Fields for multi-input windows.
WINDOW	The number of lines in the actual window frame (not the number of lines of window data). This does not include borders.
POPUP	Sets the pop-up feature. 0 This will not be a pop-up window. 1 This will be a pop-up window.

Figure 8-1. Transfer File Syntax: Window File Attributes

Attribute	Description
BORDER	Sets the window border. 0 There will be no window border. 1 There will be a window border. 2 There will be a window border. Options 1 and 2 both result in a basic window border.
HEADLEN	Length of the window heading. If this value is 0, there will be no heading.
RETURN	Sets the line break feature for use with return value display windows. 0 Line break will not be used. 1 New line before this return value. 2 New line after this return value. 3 New line before and after this value.
MULTI	Sets the multi-select feature. 0 This will not be a multi-select window. 1 This will be a multi-select window.
HEADING	The text of the window heading.
HELP	The name of the help window for this window.
HELPFILE	The name of the window file that contains the help window.

Attribute	Description
DISPLAY	The name of a window to be displayed at the same time this one is displayed. There can be up to 16 DISPLAY values for each window. This attribute is optional.
HIDE	The name of a window to be hidden when this one is displayed. There can be up to 16 HIDE values for each window. This attribute is optional.

Figure 8-2. Transfer File Syntax: Window Attributes

Attribute	Description
DATA	A line to be displayed in the window (for example, a menu choice in a vertical menu Window, or a line of text in a text display window). The data can include amper variables (including &windowname).
GOTO	The name of the window to go to if this line is selected by the user. The value can be an amper variable (including &windowname). If the value is blank, and this line is selected, Windows will return to Dialogue Manager.
VALUE	<p>The return value supplied if this line is selected by the user. This value will be placed in the amper variable &windowname, where windowname is the name of the window.</p> <p>For file names windows (TYPE = 5), this is the file selection criteria (including asterisks) of the file names to be displayed.</p> <p>For field names windows (TYPE = 6), this is the name of the Master File whose fields will be displayed.</p> <p>For file contents windows (TYPE = 7), this is the name of the file whose contents are to be displayed.</p>

Figure 8-3. Transfer File Syntax: Window Line Attributes

Operating Environment Considerations

When you transfer a window file to a mainframe operating environment from a different environment, differences in hardware and operating software may require that you make changes to the file. These changes are discussed below.

- **Screen position.** Windows should not begin in row 1 or in column 1. If you transfer a window with these row or column positions, truncation will occur. Adjust the ROW and COLUMN attributes if necessary.
- **Screen size.** Windows should not have more than 22 rows or 77 columns. Windows that extend beyond the end of the terminal screen will automatically be truncated without any warning message.

This is important to note if you are transferring a window file from an environment where the screen size differs from that in the mainframe environment. Adjust the ROW and COLUMN attributes if necessary.

- **Window Position.** Column 1 of vertical menu, horizontal menu, multi-input and text display windows cannot be used. Window text must begin to the right of column 1.
- **Function keys.** Windows transferred from other environments may refer to function keys not present in the mainframe environment. Change function key references if necessary.
- **Blank lines.** Are acknowledged by Window Painter.
- **Colors and Border Types.** The use of colored windows and background and multiple border types is not supported.
- **File Naming Conventions.** File naming conventions differ in different operating environments. When transferring a file from some environments, the Window facility will automatically translate references to FOCEXECs, Master Files, and error files, as shown below. You must change other file references yourself when you edit the transfer file.

PC or UNIX Extension	Mainframe File Type or ddname
.FEX	FOCEXEC
.MAS	MASTER
.ERR	ERRORS

Example

Sample Transfer File

To illustrate the transfer file format, part of the transfer file for the SAMPLE window file is shown below (SAMPLE is described in the tutorial). The MAIN and EXECNAME windows from the file are included in the example.

```
FILENAME=SAMPLE
DESCRIPTION='Sample file for windows tutorial'
WINDOWNAME=MAIN,TYPE=1
COMMENT='User can report, graph, or exit.'
ROW= 6,COLUMN=23,HEIGHT= 7,WIDTH=38,WINDOW= 7,POPUP= 0,BORDER= 2,HEADLEN=28
RETURN=0
MULTI=0
HEADING='Would you like to:'
DATA= '    '
$
DATA='                Create a report?'
GOTO='EXECTYPE',VALUE='RPT '
$
DATA='    '
$
DATA='                Create a graph?'
GOTO='EXECTYPE',VALUE='GRPH'
$
DATA='    '
$
DATA='                Exit?'
GOTO='                ',VALUE='XXIT'
$
DATA='    '
$
DISPLAY=BORDER      ,$
DISPLAY=BANNER      ,$
WINDOWNAME=EXECNAME,TYPE=5
COMMENT='Select an existing FOCEXEC from list.'
ROW= 4,COLUMN=11,HEIGHT=11,WIDTH=57,WINDOW=11,POPUP= 0,BORDER=
2,HEADLEN=55,
RETURN=0
MULTI=0
HEADING='Select the request you want to execute and press ENTER:'
DATA='    '
GOTO='                ',VALUE='* FOCEXEC'
$
DISPLAY=BORDER,$
HIDE=BANNER,$
HIDE=MAIN,$
HIDE=EXECTYPE,$
```

Compiling the Transfer File

The transfer file can be executed in its current format, but it may execute slowly, and it will use a large amount of memory. You can make your window application more efficient, requiring less time and memory for execution, by compiling it.

You can compile a transfer file using the WINDOW COMPILE command. This produces a new compiled window file, in the same format as the window files produced by Window Painter.

Note that before you can issue this command in MVS, a PDS with LRECL 4096 and RECFM F must have already been allocated to ddname FMU. However, you do not need to create this PDS if you are only going to use the transfer file during the current FOCUS session: Window Painter will temporarily allocate the PDS.

Syntax

How to Compile a Transfer File

```
WINDOW COMPILE windowfile
```

where:

windowfile

Is the name of the transfer file.

In CMS, this must be the file name of a file with file type TRF.

The command will create a new file with the file name specified in the command, and a file type of FMU, on the A disk. Once it has been created, you can move the file to any disk you wish.

In MVS, this must be a member name of a member of a PDS allocated to ddname TRF.

The command will create a new member of the PDS allocated to ddname FMU, with the same member name specified in the command.

When a Dialogue Manager -WINDOW command is encountered in a FOCEXEC, FOCUS will search for a compiled window file (an FMU file) with the specified file name. If the compiled file is not found, the transfer file (TRF file) with the same file name will be used.

Note that if you compile a transfer file and later make changes to it, you will need to recompile the updated transfer file: otherwise, FOCUS will continue to use the older, unchanged compiled file.

APPENDIX A

Master Files and Diagrams

Topics:

- Creating Sample Data Sources
- The EMPLOYEE Data Source
- The JOBFIL Data Source
- The EDUCFIL Data Source
- The SALES Data Source
- The PROD Data Source
- The CAR Data Source
- The LEDGER Data Source
- The FINANCE Data Source
- The REGION Data Source
- The COURSES Data Source
- The EMPDATA Data Source
- The EXPERSON Data Source
- The TRAINING Data Source
- The PAYHIST File
- The COMASTER File
- The VideoTrk and MOVIES Data Sources
- The VIDEOTR2 Data Source
- The Gotham Grinds Data Sources

This appendix contains data source descriptions and structure diagrams for the examples used throughout the documentation.

Creating Sample Data Sources

You can create the sample data sources on your user ID by executing the procedures specified below. These FOCEXECs are supplied with FOCUS. If they are not available to you or if they produce error messages, contact your systems administrator.

To create these files, first make sure you have read access to the Master Files.

Data Source	Load Procedure Name
EMPLOYEE, EDUCFILE, and JOBFILE	Under CMS enter: EX EMPTEST Under MVS, enter: EX EMPTSO These FOCEXECs also test the data sources by generating sample reports. If you are using Hot Screen, remember to press either Enter or the PF3 key after each report. If the EMPLOYEE, EDUCFILE, and JOBFILE data sources already exist on your user ID, the FOCEXEC will replace the data sources with new copies. This FOCEXEC assumes that the high-level qualifier for the FOCUS data sources will be the same as the high-level qualifier for the MASTER PDS that was unloaded from the tape.
SALES PROD	EX SALES EX PROD
CAR	none (created automatically during installation)
LEDGER FINANCE REGION COURSES EXPERSON	EX LEDGER EX FINANCE EX REGION EX COURSES EX EXPERSON
EMPDATA TRAINING	EX LOADEMP EX LOADTRAI
PAYHIST	none (PAYHIST DATA is a sequential data source and is allocated during the installation process)
COMASTER	none (COMASTER is used for debugging other Master Files)
VideoTrk and MOVIES	EX LOADVTRK
VIDEOTR2	EX LOADVID2
Gotham Grinds	EX LOADGG

The EMPLOYEE Data Source

The EMPLOYEE data source contains data about a company's employees. Its segments are:

- EMPINFO, which contains employee IDs, names, and positions.
- FUNDTRAN, which specifies employees' direct deposit accounts. This segment is unique.
- PAYINFO, which contains the employee's salary history.
- ADDRESS, which contains employees' home and bank addresses.
- SALINFO, which contains data on employees' monthly pay.
- DEDUCT, which contains data on monthly pay deductions.

The EMPLOYEE data source also contains cross-referenced segments belonging to the JOBFIL and EDUCFIL files, described later in this appendix. The segments are:

- JOBSEG (from JOBFIL), which describes the job positions held by each employee.
- SECSEG (from JOBFIL), which lists the skills required by each position.
- SKILLSEG (from JOBFIL), which specifies the security clearance needed for each job position.
- ATTNDSEG (from EDUCFIL), which lists the dates that employees attended in-house courses.
- COURSEG (from EDUCFIL), which lists the courses that the employees attended.

The EMPLOYEE Master File

```
FILENAME=EMPLOYEE, SUFFIX=FOC
SEGNAME=EMPINFO, SEGTYPE=S1
  FIELDNAME=EMP_ID, ALIAS=EID, FORMAT=A9, $
  FIELDNAME=LAST_NAME, ALIAS=LN, FORMAT=A15, $
  FIELDNAME=FIRST_NAME, ALIAS=FN, FORMAT=A10, $
  FIELDNAME=HIRE_DATE, ALIAS=HDT, FORMAT=I6YMD, $
  FIELDNAME=DEPARTMENT, ALIAS=DPT, FORMAT=A10, $
  FIELDNAME=CURR_SAL, ALIAS=CSAL, FORMAT=D12.2M, $
  FIELDNAME=CURR_JOBCODE, ALIAS=CJC, FORMAT=A3, $
  FIELDNAME=ED_HRS, ALIAS=OJT, FORMAT=F6.2, $
SEGNAME=FUNDTRAN, SEGTYPE=U, PARENT=EMPINFO
  FIELDNAME=BANK_NAME, ALIAS=BN, FORMAT=A20, $
  FIELDNAME=BANK_CODE, ALIAS=BC, FORMAT=I6S, $
  FIELDNAME=BANK_ACCT, ALIAS=BA, FORMAT=I9S, $
  FIELDNAME=EFFECT_DATE, ALIAS=EDATE, FORMAT=I6YMD, $
SEGNAME=PAYINFO, SEGTYPE=SH1, PARENT=EMPINFO
  FIELDNAME=DAT_INC, ALIAS=DI, FORMAT=I6YMD, $
  FIELDNAME=PCT_INC, ALIAS=PI, FORMAT=F6.2, $
  FIELDNAME=SALARY, ALIAS=SAL, FORMAT=D12.2M, $
  FIELDNAME=JOBCODE, ALIAS=JBC, FORMAT=A3, $
SEGNAME=ADDRESS, SEGTYPE=S1, PARENT=EMPINFO
  FIELDNAME=TYPE, ALIAS=AT, FORMAT=A4, $
  FIELDNAME=ADDRESS_LN1, ALIAS=LN1, FORMAT=A20, $
  FIELDNAME=ADDRESS_LN2, ALIAS=LN2, FORMAT=A20, $
  FIELDNAME=ADDRESS_LN3, ALIAS=LN3, FORMAT=A20, $
  FIELDNAME=ACCTNUMBER, ALIAS=ANO, FORMAT=I9L, $
SEGNAME=SALINFO, SEGTYPE=SH1, PARENT=EMPINFO
  FIELDNAME=PAY_DATE, ALIAS=PD, FORMAT=I6YMD, $
  FIELDNAME=GROSS, ALIAS=MO_PAY, FORMAT=D12.2M, $
SEGNAME=DEDUCT, SEGTYPE=S1, PARENT=SALINFO
  FIELDNAME=DED_CODE, ALIAS=DC, FORMAT=A4, $
  FIELDNAME=DED_AMT, ALIAS=DA, FORMAT=D12.2M, $
SEGNAME=JOBSEG, SEGTYPE=KU, PARENT=PAYINFO, CRFILE=JOBFILE, CRKEY=JOBCODE,$
SEGNAME=SECSEG, SEGTYPE=KLU, PARENT=JOBSEG, CRFILE=JOBFILE,$
SEGNAME=SKILLSEG, SEGTYPE=KL, PARENT=JOBSEG, CRFILE=JOBFILE,$
SEGNAME=ATTNDSEG, SEGTYPE=KM, PARENT=EMPINFO, CRFILE=EDUCFILE, CRKEY=EMP_ID,$
SEGNAME=COURSESEG, SEGTYPE=KLU, PARENT=ATTNDSEG, CRFILE=EDUCFILE,$
```

The EMPLOYEE Structure Diagram

STRUCTURE OF FOCUS FILE EMPLOYEE ON 09/15/00 AT 10.16.27

```

EMPINFO
01      S1
*****
*EMP_ID      **
*LAST_NAME   **
*FIRST_NAME  **
*HIRE_DATE   **
*            **
*****
      I
+-----+-----+-----+-----+
      I            I            I            I            I
      I FUNDTTRAN      I PAYINFO      I ADDRESS      I SALINFO      I ATTNDSEG
02      I U          03      I S1      07      I S1      08      I S1      10      I KM
*****          *****          *****          *****          .....
*BANK_NAME *      *DAT_INC **      *TYPE **      *PAY_DATE **      :DATE_ATTEND ::
*BANK_CODE *      *PCT_INC **      *ADDRESS_LN1 **      *GROSS **      :EMP_ID ::K
*BANK_ACCT *      *SALARY **      *ADDRESS_LN2 **      * **      :      :
*EFFECT_DATE *      *JOBCODE **      *ADDRESS_LN3 **      * **      :      :
* **      * **      * **      * **      :      :
*****          *****          *****          *****          :.....:
          *****          *****          *****          .....:
                        I                        I                        I EDUCFILE
                        I                        I                        I
                        I                        I                        I
                        I JOBSEG                        I DEDUCT      I COURSEG
          04      I KU          09      I S1      11      I KLU
.....          *****          .....
:JOBCODE      :K      *DED_CODE **      :COURSE_CODE :
:JOB_DESC      :      *DED_AMT **      :COURSE_NAME :
:      :      * **      :      :
:      :      * **      :      :
:      :      * **      :      :
:.....:      *****          :.....:
                        I JOBFILE          ***** EDUCFILE
                        I
+-----+-----+
      I            I
      I SECSEG      I SKILLSEG
05      I KLU      06      I KL
.....          .....
:SEC_CLEAR :      :SKILLS      :
:      :      :SKILL_DESC :
:      :      :      :
:      :      :      :
:      :      :      :
:.....:      :.....:
JOBFILE      :.....:
                        JOBFILE

```


The JOBFIL Data Source

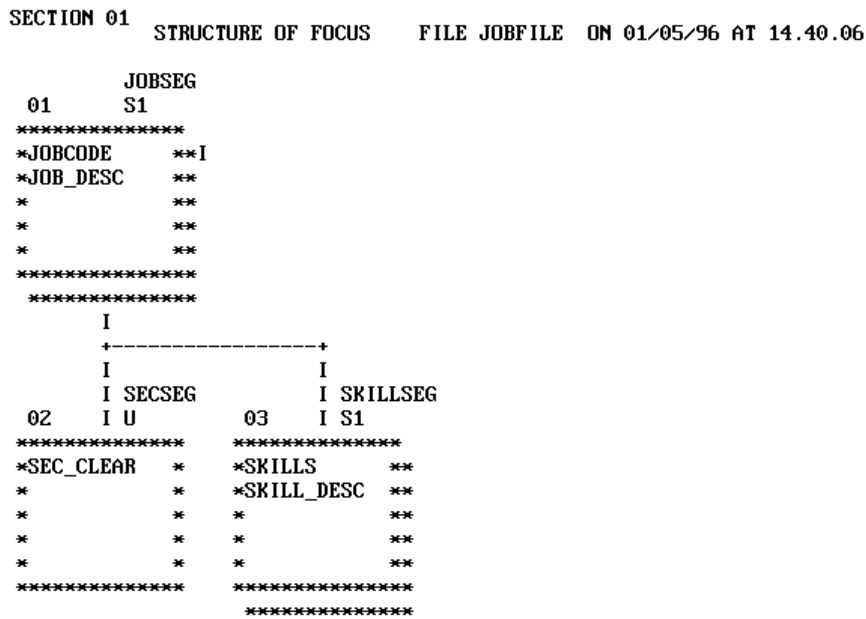
The JOBFIL data source contains information on a company’s job positions. Its segments are:

- JOBSEG describes what each position is. The field JOBCODE in this segment is indexed.
- SKILLSEG lists the skills required by each position.
- SECSEG specifies the security clearance needed, if any. This segment is unique.

The JOBFIL Master File

```
FILENAME=JOBFIL ,SUFFIX=FOC
SEGNAME=JOBSEG ,SEGTYPE=S1
FIELD=JOBCODE ,ALIAS=JC ,USAGE=A3 ,INDEX=I,$
FIELD=JOB_DESC ,ALIAS=JD ,USAGE=A25 ,,$
SEGNAME=SKILLSEG ,SEGTYPE=S1 ,PARENT=JOBSEG
FIELD=SKILLS ,ALIAS= ,USAGE=A4 ,,$
FIELD=SKILL_DESC ,ALIAS=SD ,USAGE=A30 ,,$
SEGNAME=SECSEG ,SEGTYPE=U ,PARENT=JOBSEG
FIELD=SEC_CLEAR ,ALIAS=SC ,USAGE=A6 ,,$
```

The JOBFIL Structure Diagram



The EDUCFILE Data Source

The EDUCFILE data source contains data on a company's in-house courses. Its segments are:

- COURSEG contains data on each course.
- ATTNDSEG specifies which employees attended the courses. Both fields in the segment are key fields. The field EMP_ID in this segment is indexed.

The EDUCFILE Master File

```

FILENAME=EDUCFILE ,SUFFIX=FOC
SEGNAME=COURSESEG ,SEGTYPE=S1
  FIELD=COURSE_CODE ,ALIAS=CC ,USAGE=A6 ,,$
  FIELD=COURSE_NAME ,ALIAS=CD ,USAGE=A30 ,,$
SEGNAME=ATTNDSEG ,SEGTYPE=SH2 ,PARENT=COURSESEG
  FIELD=DATE_ATTEND ,ALIAS=DA ,USAGE=I6YMD ,,$
  FIELD=EMP_ID ,ALIAS=EID ,USAGE=A9 ,INDEX=I,$

```

The EDUCFILE Structure Diagram

```

SECTION 01
      STRUCTURE OF FOCUS      FILE EDUCFILE ON 01/05/96 AT 14.45.44
      COURSEG
01      S1
*****
*COURSE_CODE **
*COURSE_NAME **
*           **
*           **
*           **
*****
      I
      I
      I
      I ATTNDSEG
02      I SH2
*****
*DATE_ATTEND **
*EMP_ID      **I
*           **
*           **
*           **
*****
*****

```

The SALES Data Source

The SALES data source records sales data for a dairy company (or a store chain). Its segments are:

- STOR_SEG lists the stores buying the products.
- DAT_SEG contains the dates of inventory.
- PRODUCT contains sales data for each product on each date. Note the following about fields in this segment:
 - The PROD_CODE field is indexed.
 - The RETURNS and DAMAGED fields have the MISSING=ON attribute.

The SALES Master File

```
FILENAME=KSALES, SUFFIX=FOC,

SEGNAME=STOR_SEG, SEGTYPE=S1,
  FIELDNAME=STORE_CODE, ALIAS=SNO,  FORMAT=A3,  $
  FIELDNAME=CITY,        ALIAS=CTY,  FORMAT=A15, $
  FIELDNAME=AREA,        ALIAS=LOC,  FORMAT=A1,  $

SEGNAME=DATE_SEG, PARENT=STOR_SEG, SEGTYPE=SH1,
  FIELDNAME=DATE,        ALIAS=DTE,  FORMAT=A4MD, $

SEGNAME=PRODUCT, PARENT=DATE_SEG, SEGTYPE=S1,
  FIELDNAME=PROD_CODE,   ALIAS=PCODE, FORMAT=A3,   FIELDTYPE=1, $
  FIELDNAME=UNIT_SOLD,   ALIAS=SOLD,  FORMAT=I5,   $
  FIELDNAME=RETAIL_PRICE, ALIAS=RP,   FORMAT=D5.2M, $
  FIELDNAME=DELIVER_AMT, ALIAS=SHIP,  FORMAT=I5,   $
  FIELDNAME=OPENING_AMT, ALIAS=INV,   FORMAT=I5,   $
  FIELDNAME=RETURNS,     ALIAS=RTN,   FORMAT=I3,   MISSING=ON, $
  FIELDNAME=DAMAGED,     ALIAS=BAD,   FORMAT=I3,   MISSING=ON, $
```

The SALES Structure Diagram

SECTION 01

STRUCTURE OF FOCUS

FILE SALES ON 01/05/96 AT 14.50.28

```

          STOR_SEG
01      S1
*****
*STORE_CODE **
*CITY      **
*AREA      **
*          **
*          **
*****
          I
          I
          I
          I DATE_SEG
02      I SH1
*****
*DATE      **
*          **I
*          **
*          **
*          **
*****
          I
          I
          I
          I PRODUCT
03      I S1
*****
*PROD_CODE **I
*UNIT_SOLD **
*RETAIL_PRICE**
*DELIVER_AMT **
*          **
*****
          I
          I

```

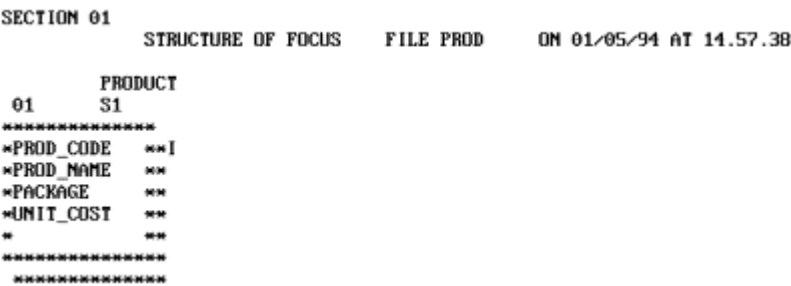
The PROD Data Source

The PROD data source lists products sold by a dairy company. It consists of one segment, PRODUCT. The field PROD_CODE is indexed.

The PROD Master File

```
FILE=KPROD, SUFFIX=FOC,  
  
SEGMENT=PRODUCT, SEGTYPE=S1,  
  FIELDNAME=PROD_CODE, ALIAS=PCODE, FORMAT=A3,   FIELDTYPE=I, $  
  FIELDNAME=PROD_NAME, ALIAS=ITEM,  FORMAT=A15,   $  
  FIELDNAME=PACKAGE,   ALIAS=SIZE,  FORMAT=A12,   $  
  FIELDNAME=UNIT_COST, ALIAS=COST,   FORMAT=D5.2M, $
```

The PROD Structure Diagram



The CAR Data Source

The CAR data source contains specifications and sales information for rare cars. Its segments are:

- ORIGIN lists the country that manufactures the car. The field COUNTRY is indexed.
- COMP contains the car name.
- CARREC contains the car model.
- BODY lists the body type, seats, dealer and retail costs, and units sold.
- SPECS lists car specifications. This segment is unique.
- WARRANT lists the type of warranty.
- EQUIP lists standard equipment.

The aliases in the CAR Master File are specified without the ALIAS keyword.

The CAR Master File

```

FILENAME=CAR,SUFFIX=FOC
SEGNAME=ORIGIN,SEGTYPE=S1
  FIELDNAME=COUNTRY,COUNTRY,A10,FIELDTYPE=I,$
SEGNAME=COMP,SEGTYPE=S1,PARENT=ORIGIN
  FIELDNAME=CAR,CARS,A16,$
SEGNAME=CARREC,SEGTYPE=S1,PARENT=COMP
  FIELDNAME=MODEL,MODEL,A24,$
SEGNAME=BODY,SEGTYPE=S1,PARENT=CARREC
  FIELDNAME=BODYTYPE,TYPE,A12,$
  FIELDNAME=SEATS,SEAT,I3,$
  FIELDNAME=DEALER_COST,DCOST,D7,$
  FIELDNAME=RETAIL_COST,RCOST,D7,$
  FIELDNAME=SALES,UNITS,I6,$
SEGNAME=SPECS,SEGTYPE=U,PARENT=BODY
  FIELDNAME=LENGTH,LEN,D5,$
  FIELDNAME=WIDTH,WIDTH,D5,$
  FIELDNAME=HEIGHT,HEIGHT,D5,$
  FIELDNAME=WEIGHT,WEIGHT,D6,$
  FIELDNAME=WHEELBASE,BASE,D6.1,$
  FIELDNAME=FUEL_CAP,FUEL,D6.1,$
  FIELDNAME=BHP,POWER,D6,$
  FIELDNAME=RPM,RPM,I5,$
  FIELDNAME=MPG,MILES,D6,$
  FIELDNAME=ACCEL,SECONDS,D6,$
SEGNAME=WARRANT,SEGTYPE=S1,PARENT=COMP
  FIELDNAME=WARRANTY,WARR,A40,$
SEGNAME=EQUIP,SEGTYPE=S1,PARENT=COMP
  FIELDNAME=STANDARD.EQUIP.A40.$

```

SECTION 01 STRUCTURE OF FOCUS FILE CAR ON 01/05/96 AT 14.59.29

```

ORIGIN
01      S1
*****
**COUNTRY      **I
**              **
**              **
**              **
**              **
*****
*****
I
I
I
I COMP
02      I S1
*****
**CAR          **
**              **
**              **
**              **
**              **
*****
*****
I
+-----+-----+-----+
I              I              I
I CARREC      I WARRANT      I EQUIP
03      I S1      06      I S1      07      I S1
*****
**MODEL        **      **WARRANTY      **      **STANDARD      **
**              **      **              **      **              **
**              **      **              **      **              **
**              **      **              **      **              **
**              **      **              **      **              **
*****
*****
I
I
I
I BODY
04      I S1
*****
**BODYTYPE     **
**SEATS        **
**DEALER_COST  **
**RETAIL_COST  **
**              **
*****
*****
I
I
I
I SPECS
05      I U
*****
**LENGTH      *
**WIDTH        *
**HEIGHT       *
**WEIGHT       *
**              *
*****

```

The LEDGER Data Source

The LEDGER data source lists accounting information. It consists of one segment, TOP. This data source is specified primarily for FML examples. Aliases do not exist for the fields in this Master File, and the commas act as placeholders.

The LEDGER Master File

```
FILENAME=LEDGER, SUFFIX=FOC,$
SEGNAME=TOP, SEGTYPE=S2,$
  FIELDNAME=YEAR    , , FORMAT=A4, $
  FIELDNAME=ACCOUNT, , FORMAT=A4, $
  FIELDNAME=AMOUNT  , , FORMAT=I5C,$
```

The LEDGER Structure Diagram

```
SECTION 01          STRUCTURE OF FOCUS   FILE LEDGER   ON 01/05/96 AT 15.07.56

          TOP
01        S2
*****
*YEAR          **
*ACCOUNT        **
*AMOUNT         **
*              **
*              **
*****
*****
```

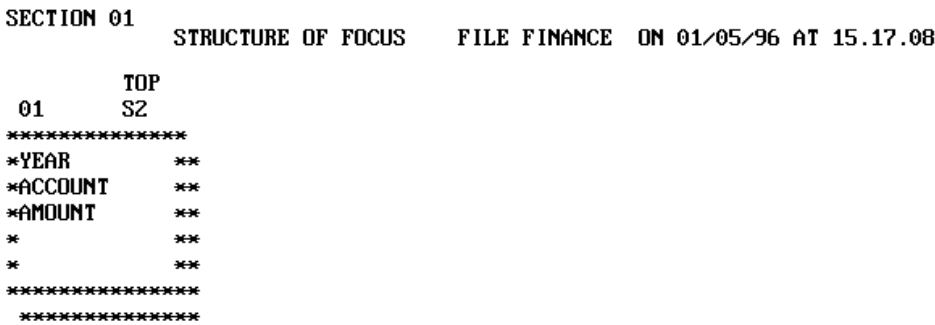

The FINANCE Data Source

The FINANCE data source contains financial information for balance sheets. It consists of one segment, TOP. This data source is specified primarily for FML examples. Aliases do not exist for the fields in this Master File, and the commas act as placeholders.

The FINANCE Master File

```
FILENAME=FINANCE, SUFFIX=FOC,$
SEGNAME=TOP, SEGTYPE=S2,$
  FIELDNAME=YEAR  , , FORMAT=A4, $
  FIELDNAME=ACCOUNT, , FORMAT=A4, $
  FIELDNAME=AMOUNT , , FORMAT=D12C,$
```

The FINANCE Structure Diagram



The REGION Data Source

The REGION data source lists account information for the east and west regions of the country. It consists of one segment, TOP. This data source is specified primarily for FML examples. Aliases do not exist for the fields in this Master File, and the commas act as placeholders.

The REGION Master File

```
FILENAME=REGION, SUFFIX=FOC,$
SEGNAME=TOP, SEGTYPE=S1,$
  FIELDNAME=ACCOUNT , , FORMAT=A4, $
  FIELDNAME=E_ACTUAL, , FORMAT=I5C,$
  FIELDNAME=E_BUDGET, , FORMAT=I5C,$
  FIELDNAME=W_ACTUAL, , FORMAT=I5C,$
  FIELDNAME=W_BUDGET, , FORMAT=I5C,$
```

The REGION Structure Diagram

```
SECTION 01      STRUCTURE OF FOCUS  FILE REGION   ON 01/05/96 AT 15.18.48

      TOP
01      S1
*****
*ACCOUNT      **
*e_ACTUAL     **
*e_BUDGET     **
*w_ACTUAL     **
*              **
*****
*****
```

The COURSES Data Source

The COURSES data source describes education courses. It consists of one segment, CRSESEG1. The field DESCRIPTION has a format of TEXT (TX).

The COURSES Master File

```
FILENAME=COURSES, SUFFIX=FOC, $
SEGNAME=CRSESEG1, SEGTYPE=S1, $
  FIELDNAME=COURSE_CODE, ALIAS=CC, FORMAT=A6, FIELDTYPE=I, $
  FIELDNAME=COURSE_NAME, ALIAS=CN, FORMAT=A30, $
  FIELDNAME=DURATION, ALIAS=DAYS, FORMAT=I3, $
  FIELDNAME=DESCRIPTION, ALIAS=CDESC, FORMAT=TX50, $
```

The COURSES Structure Diagram

```
SECTION 01          STRUCTURE OF FOCUS   FILE COURSES  ON 01/05/94 AT 15.20.59

          CRSESEG1
01        S1
*****
* COURSE_CODE **I
* COURSE_NAME **
* DURATION    **
* DESCRIPTION **T
*              **
*****
*****
```

The EMPDATA Data Source

The EMPDATA data source contains organizational data about a company's employees. It consists of one segment, EMPDATA. Note the following:

- The PIN field is indexed.
- The AREA field is a temporary one.

The EMPDATA Master File

```

FILENAME=EMPDATA, SUFFIX=FOC
SEGNAME=EMPDATA, SEGTYPE=S1
  FIELDNAME=PIN,          ALIAS=ID,          FORMAT=A9,  INDEX=I,    $
  FIELDNAME=LASTNAME,     ALIAS=LN,          FORMAT=A15,          $
  FIELDNAME=FIRSTNAME,    ALIAS=FN,          FORMAT=A10,          $
  FIELDNAME=MIDINITIAL,   ALIAS=MI,          FORMAT=A1,           $
  FIELDNAME=DIU,          ALIAS=CDIU,         FORMAT=A4,           $
  FIELDNAME=DEPT,         ALIAS=CDEPT,        FORMAT=A20,          $
  FIELDNAME=JOBCLASS,     ALIAS=CJCLAS,       FORMAT=A8,           $
  FIELDNAME=TITLE,        ALIAS=CFUNC,         FORMAT=A20,          $
  FIELDNAME=SALARY,       ALIAS=CSAL,         FORMAT=D12.2M,       $
  FIELDNAME=HIREDATE,     ALIAS=HDAT,         FORMAT=YMD,          $
$
DEFINE AREA/A13=DECODE DIU (NE 'NORTH EASTERN' SE 'SOUTH EASTERN'
CE 'CENTRAL' WE 'WESTERN' CORP 'CORPORATE' ELSE 'INVALID AREA');$

```

The EMPDATA Structure Diagram

```

SECTION 01      STRUCTURE OF FOCUS  FILE EMPDATA  ON 01/05/96 AT 14.49.09

      EMPDATA
01      S1
*****
*PIN          **I
*LASTNAME     **
*FIRSTNAME    **
*MIDINITIAL   **
*             **
*****
*****

```

The EXPERSON Data Source

The EXPERSON data source contains personal data about individual employees. It consists of one segment, ONESEG.

The EXPERSON Master File

```
FILE=EXPERSON      ,SUFFIX=FOC
SEGMENT=ONESEG, $
  FIELDNAME=SOC_SEC_NO    ,ALIAS=SSN           ,USAGE=A9      ,,$
  FIELDNAME=FIRST_NAME    ,ALIAS=FN           ,USAGE=A9      ,,$
  FIELDNAME=LAST_NAME     ,ALIAS=LN           ,USAGE=A10     ,,$
  FIELDNAME=AGE           ,ALIAS=YEARS        ,USAGE=I2      ,,$
  FIELDNAME=SEX           ,ALIAS=             ,USAGE=A1      ,,$
  FIELDNAME=MARITAL_STAT  ,ALIAS=MS           ,USAGE=A1      ,,$
  FIELDNAME=NO_DEP        ,ALIAS=NDP          ,USAGE=I3      ,,$
  FIELDNAME=DEGREE        ,ALIAS=             ,USAGE=A3      ,,$
  FIELDNAME=NO_CARS       ,ALIAS=CARS         ,USAGE=I3      ,,$
  FIELDNAME=ADDRESS       ,ALIAS=             ,USAGE=A14     ,,$
  FIELDNAME=CITY          ,ALIAS=             ,USAGE=A10     ,,$
  FIELDNAME=WAGE          ,ALIAS=PAY          ,USAGE=D10.2SM ,,$
  FIELDNAME=CATEGORY      ,ALIAS=STATUS       ,USAGE=A1      ,,$
  FIELDNAME=SKILL_CODE    ,ALIAS=SKILLS       ,USAGE=A5      ,,$
  FIELDNAME=DEPT_CODE     ,ALIAS=WHERE        ,USAGE=A4      ,,$
  FIELDNAME=TEL_EXT       ,ALIAS=EXT          ,USAGE=I4      ,,$
  FIELDNAME=DATE_EMP      ,ALIAS=BASE_DATE    ,USAGE=I6YMTD  ,,$
  FIELDNAME=MULTIPLIER    ,ALIAS=RATIO        ,USAGE=D5.3    ,,$
```

The EXPERSON Structure Diagram

```
SECTION 01          STRUCTURE OF FOCUS   FILE EXPERSON ON 01/05/96 AT 14.50.58

      ONESEG
01      S1
*****
*SOC_SEC_NO  **
*FIRST_NAME **
*LAST_NAME  **
*AGE        **
*           **
*****
*****
```

The TRAINING Data Source

The TRAINING data source contains training course data for employees. It consists of one segment, TRAINING. Note the following:

- The PIN field is indexed.
- The EXPENSES, GRADE, and LOCATION fields have the MISSING=ON attribute.

The TRAINING Master File

```

FILENAME=TRAINING, SUFFIX=FOC
SEGNAME=TRAINING, SEGTYPE=SH3
FIELDNAME=PIN,          ALIAS=ID,          FORMAT=A9,          INDEX=I,          $
FIELDNAME=COURSESTART,  ALIAS=CSTART,  FORMAT=YMD,          $
FIELDNAME=COURSECODE,   ALIAS=CCOD,   FORMAT=A7,          $
FIELDNAME=EXPENSES,     ALIAS=COST,     FORMAT=D8.2,  MISSING=ON,$
FIELDNAME=GRADE,        ALIAS=GRA,        FORMAT=A2,          MISSING=ON,$
FIELDNAME=LOCATION,      ALIAS=LOC,      FORMAT=A6,          MISSING=ON,$

```

The TRAINING Structure Diagram

```

SECTION 01
          STRUCTURE OF FOCUS    FILE TRAINING ON 12/12/94 AT 14.51.28

          TRAINING
01      SH3
*****
*PIN          **I
*COURSESTART **
*COURSECODE   **
*EXPENSES     **
*             **
*****
*****

```

The PAYHIST File

The PAYHIST data source contains the employees’ salary history. It consists of one segment, PAYSEG. The SUFFIX attribute indicates that the data file is a fixed-format sequential file.

The PAYHIST Master File

```
FILENAME=PAYHIST, SUFFIX=FIX
SEGMENT=PAYSEG,$
  FIELDNAME=SOC_SEC_NO, ALIAS=SSN, USAGE=A9, ACTUAL=A9,$
  FIELDNAME=DATE_OF_IN, ALIAS=INCDATE, USAGE=I6YMTD, ACTUAL=A6,$
  FIELDNAME=AMT_OF_INC, ALIAS=RAISE, USAGE=D6.2, ACTUAL=A10,$
  FIELDNAME=PCT_INC, ALIAS=, USAGE=D6.2, ACTUAL=A6,$
  FIELDNAME=NEW_SAL, ALIAS=CURR_SAL, USAGE=D10.2, ACTUAL=A11,$
  FIELDNAME=FILL, ALIAS=, USAGE=A38, ACTUAL=A38,$
```

The PAYHIST Structure Diagram

```
SECTION 01          STRUCTURE OF FIX    FILE PAYHIST  ON 01/05/96 AT 14.51.59

          PAYSEG
01          S1
*****
*SOC_SEC_NO **
*DATE_OF_IN **
*AMT_OF_INC **
*PCT_INC    **
*           **
*****
*****
```

The COMASTER File

The COMASTER file is used to display the file structure and contents of each segment in a data source. Since COMASTER is used for debugging other Master Files, a corresponding FOCEXEC does not exist for the COMASTER file. Its segments are:

- FILEID lists file information.
- RECID lists segment information.
- FIELDID lists field information.
- DEFREC lists a description record.
- PASSREC lists read/write access.
- CRSEG lists cross-reference information for segments.
- ACCSEG lists DBA information.

The COMASTER Master File

```

FILE=COMASTER, SUFFIX=COM,

SEGNAME=FILEID
FIELDNAME=FILENAME ,FILE ,A8 , ,,$
FIELDNAME=FILE SUFFIX ,SUFFIX ,A8 , ,,$

SEGNAME=RECID
FIELDNAME=SEGNAME ,SEGMENT ,A8 , ,,$
FIELDNAME=SEGTYPE ,SEGTYPE ,A4 , ,,$
FIELDNAME=SEGSIZE ,SEGSIZE ,14 , A4,$
FIELDNAME=PARENT ,PARENT ,A8 , ,,$
FIELDNAME=CRKEY ,CRKEY ,A66, ,,$

SEGNAME=FIELDID
FIELDNAME=FIELDNAME ,FIELD ,A66, ,,$
FIELDNAME=ALIAS ,SYNONYM ,A66, ,,$
FIELDNAME=FORMAT ,USAGE ,A8 , ,,$
FIELDNAME=ACTUAL ,ACTUAL ,A8 , ,,$
FIELDNAME=AUTHORITY ,AUTHCODE ,A8 , ,,$
FIELDNAME=FIELDTYPE ,INDEX ,A8 , ,,$
FIELDNAME=TITLE ,TITLE ,A64, ,,$
FIELDNAME=HELPMESSAGE,MESSAGE ,A256, ,,$
FIELDNAME=MISSING ,MISSING ,A4, ,,$
FIELDNAME=ACCEPTS ,ACCEPTABLE ,A255, ,,$
FIELDNAME=RESERVED ,RESERVED ,A44, ,,$

SEGNAME=DEFREC
FIELDNAME=DEFINITION ,DESCRIPTION,A44, ,,$

SEGNAME=PASSREC,PARENT=FILEID
FIELDNAME=READ/WRITE ,RW ,A32, ,,$

SEGNAME=CRSEG,PARENT=RECID
FIELDNAME=CRFILENAME ,CRFILE ,A8 , ,,$
FIELDNAME=CRSEGNAME ,CRSEGMENT ,A8 , ,,$
FIELDNAME=ENCRYPT ,ENCRYPT ,A4 , ,,$

SEGNAME=ACCSEG,PARENT=DEFREC
FIELDNAME=DBA ,DBA ,A8 , ,,$
FIELDNAME=DBAFILE , ,A8 , ,,$
FIELDNAME=USER ,PASS ,A8 , ,,$
FIELDNAME=ACCESS ,ACCESS ,A8 , ,,$
FIELDNAME=RESTRICT ,RESTRICT ,A8 , ,,$
FIELDNAME=NAME ,NAME ,A66, ,,$
FIELDNAME=VALUE ,VALUE ,A80, ,,$

```

The COMASTER Structure Diagram

SECTION 01

STRUCTURE OF EXTERNAL FILE COMASTER ON 12/12/94 AT 14.53.38

```

      FILEID
01      S1
*****
*FILENAME **
*FILE SUFFIX **
*      **
*      **
*      **
*****
*****
      I
      +-----+
      I          I
      I RECID      I PASSREC
02      I N      07      I N
*****          *****
*SEGNAME **      *READ/WRITE **
*SEGTYP      **      *      **
*SEGSIZE      **      *      **
*PARENT      **      *      **
*      **      *      **
*****          *****
*****          *****
      I
      +-----+
      I          I
      I FIELDID      I CRSEG
03      I N      06      I N
*****          *****
*FIELDNAME **      *CRFILENAME **
*ALIAS      **      *CRSEGNAME **
*FORMAT      **      *ENCRYPT      **
*ACTUAL      **      *      **
*      **      *      **
*****          *****
*****          *****
      I
      I
      I
      I DEFREC
04      I N
*****
*DEFINITION **
*      **
*      **
*      **
*      **
*****
*****
      I
      I
      I
      I ACCSEG
05      I N
*****
*DBA      **
*DBAFILE      **
*USER      **
*ACCESS      **
*      **
*****
*****

```

The VideoTrk and MOVIES Data Sources

The VideoTrk data source tracks customer, rental, and purchase information for a video rental business. It can be joined to the MOVIES data source. VideoTrk and MOVIES are used in examples that illustrate the use of the Maintain facility.

VideoTrk Master File

```
FILENAME=VIDEOTRK,  SUFFIX=FOC
SEGNAME=CUST,      SEGTYPE=S1
    FIELDNAME=CUSTID,    ALIAS=CIN,        FORMAT=A4,  $
    FIELDNAME=LASTNAME,  ALIAS=LN,        FORMAT=A15, $
    FIELDNAME=FIRSTNAME, ALIAS=FN,        FORMAT=A10, $
    FIELDNAME=EXPDATE,   ALIAS=EXDAT,      FORMAT=YMD, $
    FIELDNAME=PHONE,     ALIAS=TEL,        FORMAT=A10, $
    FIELDNAME=STREET,    ALIAS=STR,        FORMAT=A20, $
    FIELDNAME=CITY,      ALIAS=CITY,       FORMAT=A20, $
    FIELDNAME=STATE,     ALIAS=PROV,       FORMAT=A4,  $
    FIELDNAME=ZIP,       ALIAS=POSTAL_CODE, FORMAT=A9,  $
SEGNAME=TRANSDAT,    SEGTYPE=SH1,  PARENT=CUST
    FIELDNAME=TRANSDATE, ALIAS=OUTDATE,    FORMAT=YMD, $
SEGNAME=SALES,       SEGTYPE=S2,    PARENT=TRANSDAT
    FIELDNAME=PRODCODE,  ALIAS=PCOD,       FORMAT=A6,  $
    FIELDNAME=TRANSCODE, ALIAS=TCOD,       FORMAT=I3,  $
    FIELDNAME=QUANTITY,  ALIAS=NO,         FORMAT=I3S, $
    FIELDNAME=TRANSTOT,  ALIAS=TTOT,       FORMAT=F7.2S, $
SEGNAME=RENTALS,     SEGTYPE=S2,    PARENT=TRANSDAT
    FIELDNAME=MOVIECODE, ALIAS=MCOD,       FORMAT=A6,  INDEX=I, $
    FIELDNAME=COPY,      ALIAS=COPY,       FORMAT=I2,  $
    FIELDNAME=RETURNDATE, ALIAS=INDATE,    FORMAT=YMD, $
    FIELDNAME=FEE,       ALIAS=FEE,        FORMAT=F5.2S, $
```

MOVIES Master File

```
FILENAME=MOVIES,    SUFFIX=FOC
SEGNAME=MOVINFO,    SEGTYPE=S1
    FIELDNAME=MOVIECODE, ALIAS=MCOD,       FORMAT=A6,  INDEX=I, $
    FIELDNAME=TITLE,     ALIAS=MTL,        FORMAT=A39, $
    FIELDNAME=CATEGORY,  ALIAS=CLASS,      FORMAT=A8,  $
    FIELDNAME=DIRECTOR,  ALIAS=DIR,        FORMAT=A17, $
    FIELDNAME=RATING,    ALIAS=RTG,        FORMAT=A4,  $
    FIELDNAME=RELDATE,   ALIAS=RDAT,       FORMAT=YMD, $
    FIELDNAME=WHOLESALEPR, ALIAS=WPRC,     FORMAT=F6.2, $
    FIELDNAME=LISTPR,    ALIAS=LPRC,       FORMAT=F6.2, $
    FIELDNAME=COPIES,    ALIAS=NOC,        FORMAT=I3,  $
```

VideoTrk Structure Diagram

SECTION 01

STRUCTURE OF FOCUS

FILE VIDEOTRK ON 05/21/99 AT 12.25.19

```

          CUST
01      S1
*****
*CUSTID      **
*LASTNAME    **
*FIRSTNAME   **
*EXPDATE     **
*            **
*****
          I
          I
          I
          I  TRANSDAT
02      I SH1
*****
*TRANSDATE   **
*            **
*            **
*            **
*            **
*****
          I
          +-----+
          I              I
          I  SALES      I  RENTALS
03      I S2          04  I S2
*****              *****
*PRODCODE    **    *MOVIECODE  **I
*TRANSCODE   **    *COPY       **
*QUANTITY    **    *RETURNDATE **
*TRANSTOT    **    *FEE        **
*            **    *            **
*****              *****
          *****

```

MOVIES Structure Diagram

```
SECTION 01
STRUCTURE OF FOCUS FILE MOVIES ON 05/21/99 AT 12.26.05

MOVINFO
01 S1
*****
*MOVIECODE **I
*TITLE **
*CATEGORY **
*DIRECTOR **
* **
*****
*****
```

The VIDEOTR2 Data Source

The VIDEOTR2 data source tracks customer, rental, and purchase information for a video rental business. It is similar to VideoTrk but is a partitioned data source with both a Master and Access File and with a date-time field.

The VIDEOTR2 Master File

```
FILENAME=VIDEOTR2, SUFFIX=FOC,
ACCESS=VIDEOACX, $
SEGNAME=CUST, SEGTYPE=S1
FIELDNAME=CUSTID, ALIAS=CIN, FORMAT=A4, $
FIELDNAME=LASTNAME, ALIAS=LN, FORMAT=A15, $
FIELDNAME=FIRSTNAME, ALIAS=FN, FORMAT=A10, $
FIELDNAME=EXPDATE, ALIAS=EXDAT, FORMAT=YMD, $
FIELDNAME=PHONE, ALIAS=TEL, FORMAT=A10, $
FIELDNAME=STREET, ALIAS=STR, FORMAT=A20, $
FIELDNAME=CITY, ALIAS=CITY, FORMAT=A20, $
FIELDNAME=STATE, ALIAS=PROV, FORMAT=A4, $
FIELDNAME=ZIP, ALIAS=POSTAL_CODE, FORMAT=A9, $
FIELDNAME=EMAIL, ALIAS=EMAIL, FORMAT=A18, $
SEGNAME=TRANSDAT, SEGTYPE=SH1, PARENT=CUST
FIELDNAME=TRANSDATE, ALIAS=OUTDATE, FORMAT=HYYYMDI, $
SEGNAME=SALES, SEGTYPE=S2, PARENT=TRANSDAT
FIELDNAME=TRANSCODE, ALIAS=TCOD, FORMAT=I3, $
FIELDNAME=QUANTITY, ALIAS=NO, FORMAT=I3S, $
FIELDNAME=TRANSTOT, ALIAS=TTOT, FORMAT=F7.2S, $
SEGNAME=RENTALS, SEGTYPE=S2, PARENT=TRANSDAT
FIELDNAME=MOVIECODE, ALIAS=MCOD, FORMAT=A6, INDEX=I, $
FIELDNAME=COPY, ALIAS=COPY, FORMAT=I2, $
FIELDNAME=RETURNDATE, ALIAS=INDATE, FORMAT=YMD, $
FIELDNAME=FEE, ALIAS=FEE, FORMAT=F5.2S, $
DEFINE DATE/I4 = HPART(TRANSDATE, 'YEAR', 'I4');
```

The VIDEOTR2 Access File

On CMS,

```
MASTER VIDEOTR2
  DATANAME 'VIDPART1 FOCUS A'
  WHERE DATE EQ 1991;

  DATANAME 'VIDPART2 FOCUS A'
  WHERE DATE FROM 1996 TO 1998;

  DATANAME 'VIDPART3 FOCUS A'
  WHERE DATE FROM 1999 TO 2000;
```

On MVS, the data set names include your user ID as the high-level qualifier:

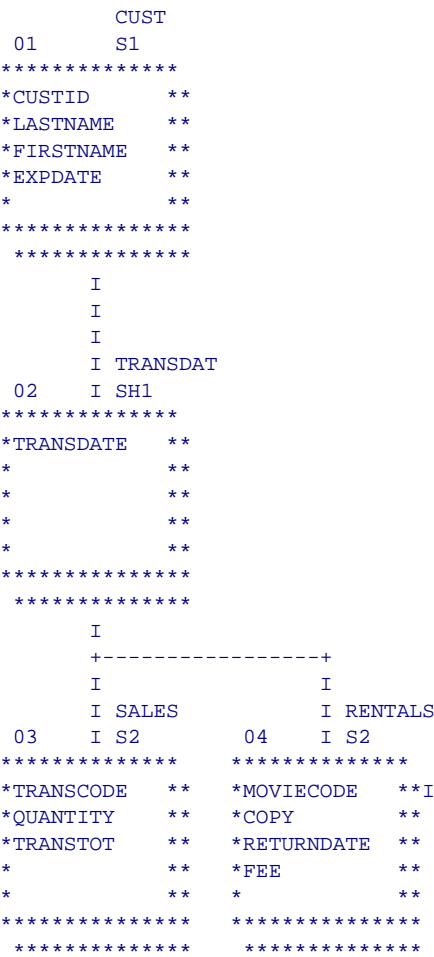
```
MASTER VIDEOTR2
  DATANAME userid.VIDPART1.FOCUS
  WHERE DATE EQ 1991;

  DATANAME userid.VIDPART2.FOCUS
  WHERE DATE FROM 1996 TO 1998;

  DATANAME userid.VIDPART2.FOCUS
  WHERE DATE FROM 1999 TO 2000;
```

The VIDEOTR2 Structure Diagram

STRUCTURE OF FOCUS FILE VIDEOTR2 ON 09/27/00 AT 16.45.48



The Gotham Grinds Data Sources

Gotham Grinds is a group of data sources that contain information about a specialty items company.

The GGDEMOG Data Source

The GGDEMOG data source contains demographic information about the customers of Gotham Grinds, a company that sells specialty items like coffee, gourmet snacks, and gifts. It consists of one segment, DEMOG01.

The GGDEMOG Master File

```
FILENAME=GGDEMOG, SUFFIX=FOC
SEGNAME=DEMOG01, SEGTYPE=S1
  FIELD=ST, ALIAS=E02, FORMAT=A02, INDEX=I, TITLE='State', DESC='State', $
  FIELD=HH, ALIAS=E03, FORMAT=I09, TITLE='Number of Households',
    DESC='Number of Households', $
  FIELD=AUGHHS298, ALIAS=E04, FORMAT=I09, TITLE='Average Household Size',
    DESC='Average Household Size', $
  FIELD=MEDHHI98, ALIAS=E05, FORMAT=I09, TITLE='Median Household Income',
    DESC='Median Household Income', $
  FIELD=AUGHHI98, ALIAS=E06, FORMAT=I09, TITLE='Average Household Income',
    DESC='Average Household Income', $
  FIELD=MALEPOP98, ALIAS=E07, FORMAT=I09, TITLE='Male Population',
    DESC='Male Population', $
  FIELD=FEMPOP98, ALIAS=E08, FORMAT=I09, TITLE='Female Population',
    DESC='Female Population', $
  FIELD=P15T01998, ALIAS=E09, FORMAT=I09, TITLE='15 to 19',
    DESC='Population 15 to 19 years old', $
  FIELD=P20T02998, ALIAS=E10, FORMAT=I09, TITLE='20 to 29',
    DESC='Population 20 to 29 years old', $
  FIELD=P30T04998, ALIAS=E11, FORMAT=I09, TITLE='30 to 49',
    DESC='Population 30 to 49 years old', $
  FIELD=P50T06498, ALIAS=E12, FORMAT=I09, TITLE='50 to 64',
    DESC='Population 50 to 64 years old', $
  FIELD=P65OVR98, ALIAS=E13, FORMAT=I09, TITLE='65 and over',
    DESC='Population 65 and over', $
```


The GGDEMOG Structure Diagram

```

NUMBER OF ERRORS=      0
NUMBER OF SEGMENTS=    1 ( REAL=    1 VIRTUAL=    0 )
NUMBER OF FIELDS=     12 INDEXES=    1 FILES=      1
TOTAL LENGTH OF ALL FIELDS= 101
SECTION 01
                STRUCTURE OF FOCUS      FILE GGDEMOG  ON 09/17/96 AT 12.18.05

                GGDEMOG
01             S1
*****
*ST             **I
*HH             **
*AVGHHSZ98      **
*MEDHHI98       **
*               **
*****
*****

```

The GGORDER Data Source

The GGORDER data source contains order information for Gotham Grinds. It consists of two segments, ORDER01 and ORDER02, respectively.

The GGORDER Master File

```

FILENAME=GGORDER, SUFFIX=FOC
SEGNAME=ORDER01, SEGTYPE=S1
  FIELD=ORDER_NUMBER, ALIAS=ORDNO1, FORMAT=I6, TITLE='Order,Number',
    DESC='Order Identification Number', $
  FIELD=ORDER_DATE, ALIAS=DATE, FORMAT=MDY, TITLE='Order,Date',
    DESC='Date order was placed', $
  FIELD=STORE_CODE, ALIAS=STCD, FORMAT=A5, TITLE='Store,Code',
    DESC='Store Identification Code (for order)', $
  FIELD=PRODUCT_CODE, ALIAS=PCD, FORMAT=A4, TITLE='Product,Code',
    DESC='Product Identification Code (for order)', $
  FIELD=QUANTITY, ALIAS=ORDUNITS, FORMAT=I8, TITLE='Ordered,Units',
    DESC='Quantity Ordered', $
SEGNAME=ORDER02, SEGTYPE=KU, PARENT=ORDER01, CRFILE=GGPRODS, CRKEY=PCD,
CRSEG=PRODS01 , $

```

The GGORDER Structure Diagram

```

NUMBER OF ERRORS=      0
NUMBER OF SEGMENTS=    2  ( REAL=    1  VIRTUAL=    1 )
NUMBER OF FIELDS=     12  INDEXES=    0  FILES=    2
TOTAL LENGTH OF ALL FIELDS=    92
SECTION 01
      STRUCTURE OF FOCUS      FILE GGORDER  ON 09/17/96 AT 12.29.24

      GGORDER
01      S1
*****
*ORDER_NUMBER**
*ORDER_DATE **
*STORE_CODE **
*PRODUCT_CODE**
*      **
*****
      I
      I
      I
      I ORDER02
02      I KU
-----
:PRODUCT_ID :K
:PRODUCT_DES>:
:VENDOR_CODE :
:VENDOR_NAME :
:      :
:-----:

```

The GGPRODS Data Source

The GGPRODS data source contains product information for Gotham Grinds. It consists of one segment, PRODS01.

The GGPRODS Master File

```

FILENAME=GGPRODS, SUFFIX=FOC
SEGNAME=PRODS01, SEGTYPE=S1
FIELD=PRODUCT_ID, ALIAS=PCD, FORMAT=A4, INDEX=I, TITLE='Product,Code',
DESC='Product Identification Code', $
FIELD=PRODUCT_DESCRIPTION, ALIAS=PRODUCT, FORMAT=A16, TITLE='Product',
DESC='Product Name', $
FIELD=VENDOR_CODE, ALIAS=UCD, FORMAT=A4, INDEX=I, TITLE='Vendor ID',
DESC='Vendor Identification Code', $
FIELD=VENDOR_NAME, ALIAS=VENDOR, FORMAT=A23, TITLE='Vendor Name',
DESC='Vendor Name', $
FIELD=PACKAGE_TYPE, ALIAS=PACK, FORMAT=A7, TITLE='Package',
DESC='Packaging Style', $
FIELD=SIZE, ALIAS=SZ, FORMAT=I2, TITLE='Size', DESC='Package Size', $
FIELD=UNIT_PRICE, ALIAS=UNITPR, FORMAT=D7.2, TITLE='Unit,Price',
DESC='Price for one unit', $

```

The GGPRODS Structure Diagram

```
NUMBER OF ERRORS=          0
NUMBER OF SEGMENTS=        1 ( REAL=      1 VIRTUAL=    0 )
NUMBER OF FIELDS=          7 INDEXES=      2 FILES=        1
TOTAL LENGTH OF ALL FIELDS=    63
SECTION 01
        STRUCTURE OF FOCUS   FILE GGPRODS   ON 09/17/96 AT 12.21.12

        GGPRODS
01      S1
*****
*PRODUCT_ID   **I
*VENDOR_CODE  **I
*PRODUCT_DES>***
*VENDOR_NAME  **
*              **
*****
```

The GGSales Data Source

The GGSales data source contains sales information for Gotham Grinds. It consists of one segment, SALES01.

The GGSales Master File

```
FILENAME=GGSales, SUFFIX=FOC
SEGNAME=SALES01, SEGTYPE=S1
FIELD=SEQ_NO, ALIAS=SEQ, FORMAT=I5, TITLE='Sequence#',
DESC='Sequence number in database',$
FIELD=CATEGORY, ALIAS=E02, FORMAT=A11, INDEX=I, TITLE='Category',
DESC='Product category',$
FIELD=PCD, ALIAS=E03, FORMAT=A04, INDEX=I, TITLE='Product ID',
DESC='Product Identification code (for sale)',$
FIELD=PRODUCT, ALIAS=E04, FORMAT=A16, TITLE='Product', DESC='Product name',$
FIELD=REGION, ALIAS=E05, FORMAT=A11, INDEX=I, TITLE='Region',
DESC='Region code',$
FIELD=ST, ALIAS=E06, FORMAT=A02, INDEX=I, TITLE='State', DESC='State',$
FIELD=CITY, ALIAS=E07, FORMAT=A20, TITLE='City', DESC='City',$
FIELD=STCD, ALIAS=E08, FORMAT=A05, INDEX=I, TITLE='Store ID',
DESC='Store identification code (for sale)',$
FIELD=DATE, ALIAS=E09, FORMAT=I8YYMD, TITLE='Date',
DESC='Date of sales report',$
FIELD=UNITS, ALIAS=E10, FORMAT=I08, TITLE='Unit Sales',
DESC='Number of units sold',$
FIELD=DOLLARS, ALIAS=E11, FORMAT=I08, TITLE='Dollar Sales',
DESC='Total dollar amount of reported sales',$
FIELD=BUDUNITS, ALIAS=E12, FORMAT=I08, TITLE='Budget Units',
DESC='Number of units budgeted',$
FIELD=BUDDOLLARS, ALIAS=E13, FORMAT=I08, TITLE='Budget Dollars',
DESC='Total sales quota in dollars',$
```

The GGSales Structure Diagram

```

NUMBER OF ERRORS=      0
NUMBER OF SEGMENTS=    1 ( REAL=    1 VIRTUAL=    0 )
NUMBER OF FIELDS=     13 INDEXES=    5 FILES=    1
TOTAL LENGTH OF ALL FIELDS= 114
SECTION 01
                STRUCTURE OF FOCUS    FILE GGSales  ON 09/17/96 AT 12.22.19

                GGSales
01             S1
*****
*SEQ_NO       **
*CATEGORY     **I
*PCD          **I
*REGION       **I
*             **
*****
*****

```

The GGSTORES Data Source

The GGSTORES data source contains information for each of Gotham Grinds' 12 stores in the United States. It consists of one segment, STORES01.

The GGSTORES Master File

```

FILENAME=GGSTORES, SUFFIX=FOC
SEGNAME=STORES01, SEGTYPE=S1
  FIELD=STORE_CODE, ALIAS=E02, FORMAT=A05, INDEX=I, TITLE='Store ID',
    DESC='Franchisee ID Code', $
  FIELD=STORE_NAME, ALIAS=E03, FORMAT=A23, TITLE='Store Name',
    DESC='Store Name', $
  FIELD=ADDRESS1, ALIAS=E04, FORMAT=A19, TITLE='Contact',
    DESC='Franchisee Owner', $
  FIELD=ADDRESS2, ALIAS=E05, FORMAT=A31, TITLE='Address', DESC='Street Address', $
  FIELD=CITY, ALIAS=E06, FORMAT=A22, TITLE='City', DESC='City', $
  FIELD=STATE, ALIAS=E07, FORMAT=A02, INDEX=I, TITLE='State', DESC='State', $
  FIELD=ZIP, ALIAS=E08, FORMAT=A06, TITLE='Zip Code', DESC='Postal Code', $

```

The GGSTORES Structure Diagram

```
NUMBER OF ERRORS=      0
NUMBER OF SEGMENTS=    1 ( REAL=    1 VIRTUAL=    0 )
NUMBER OF FIELDS=      7 INDEXES=    2 FILES=      1
TOTAL LENGTH OF ALL FIELDS= 108
```

SECTION 01

STRUCTURE OF FOCUS FILE GGSTORES ON 09/17/96 AT 12.23.09

```
                GGSTORES
01             S1
*****
*STORE_CODE   **I
*STATE_       **I
*STORE_NAME   **
*ADDRESS1     **
*             **
*****
*****
```

APPENDIX B

Error Messages

Topics:

- Accessing Error Files
- Displaying Messages Online

If you need to see the text or explanation for any error message, you can display it online in your FOCUS session or find it in a standard FOCUS ERRORS file. All of the FOCUS error messages are stored in eight system ERRORS files.

Accessing Error Files

For CMS, the ERRORS files are:

- FOT004 ERRORS
- FOG004 ERRORS
- FOM004 ERRORS
- FOS004 ERRORS
- FOA004 ERRORS
- FSQXLXT ERRORS
- FOCSTY ERRORS
- FOB004 ERRORS

For MVS, these files are the following members in the ERRORS PDS:

- FOT004
- FOG004
- FOM004
- FOS004
- FOA004
- FSQXLXT
- FOCSTY
- FOB004

Displaying Messages Online

To display a message online, issue the following query command at the FOCUS command level

? *n*

where *n* is the message number.

The message number and text will display along with a detailed explanation of the message (if one exists). For example, issuing the following command

? 210

displays the following:

(FOC210) THE DATA VALUE HAS A FORMAT ERROR:

An alphabetic character has been found where all numerical digits are required.

Index

Symbols

-“...” command, 3-10, 3-96

&ACCEPTS variable, 3-59

&BASEIO variable, 3-59

&CHNGD variable, 3-59

&CURSOR variable, 3-61

&CURSORAT variable, 3-61

&DATE variable, 3-54, 3-57

&DATEfmt variable, 3-54, 6-31

&DELTD variable, 3-59

&DMY variable, 3-54

&DMYY variable, 3-54, 3-58, 6-31

&DUPLS variable, 3-59

&ECHO variable, 3-42, 3-61

&FOCCPU variable, 3-54

&FOCDISORG variable, 2-10, 3-59

&FOCERRNUM variable, 3-59

&FOCEXTTRM variable, 3-54

&FOCFIELDNAME variable, 3-54

&FOCFOCEXEC variable, 3-55, 3-57

&FOCINCLUDE variable, 3-55

&FOCMODE variable, 3-55

&FOCPRINT variable, 3-55

&FOCPUTLVL variable, 3-55

&FOCQUALCHAR variable, 3-55

&FOCREL variable, 3-55

&FOCSBORDER variable, 3-55

&FOCSYSTYP variable, 3-55

&FOCTMPDSK variable, 3-56

&FOCTRMSD variable, 3-56

&FOCTRMSW variable, 3-56

&FOCTRMTYP variable, 3-56

&FOCTTIME variable, 3-56

&FOCVTIME variable, 3-56

&FORMAT variable, 3-59

&HIPERFOCUS variable, 3-56

&INPUT variable, 3-59

&INVALID variable, 3-59

&IORETURN variable, 3-56

&LINES variable, 3-59 to 3-60

&MDY variable, 3-56

&MDYY variable, 3-56, 3-58, 6-31

&myvar, 3-10

&NOMATCH variable, 3-60

&PFKEY variable, 3-61, 8-26

&QUIT variable, 3-44, 3-61

&READS variable, 3-60

&RECORDS variable, 3-60

&REJECTS variable, 3-60

&RETCODE variable, 3-23, 3-56

&STACK variable, 3-43, 3-61

&TOD variable, 3-56

&TRANS variable, 3-60

&WINDOWNAME variable, 3-61, 8-26

&WINDOWVALUE variable, 3-61, 8-26

&YMD variable, 3-56

&YYMD variable, 3-56, 3-58, 6-31

- * command, 3-8 to 3-9, 3-80
- .EVAL operator, 3-62 to 3-63
- .EXIST operator, 3-20
- .TYPE operator, 3-22
- ? && command, 2-2, 2-27
- ? &[string] command, 3-10, 3-51
- ? COMBINE command, 2-2 to 2-3
- ? command, 3-80
- ? DEFINE command, 2-2, 2-4
- ? EUROFILE command, 2-2, 2-5, 7-14
- ? F command, 2-2, 2-5
- ? FDT command, 2-2, 2-6 to 2-7
- ? FF command, 2-2, 2-8
- ? FILE command, 2-2, 2-9 to 2-10
- ? FUNCTION command, 2-2, 2-11
- ? HBUDGET command, 2-2, 2-11 to 2-12
- ? HOLD command, 2-2, 2-12
- ? JOIN command, 2-2, 2-13
- ? LANG command, 2-2, 2-14
- ? LET command, 2-2, 2-14, 4-10
- ? LOAD command, 2-2, 2-15, 5-6
- ? n command, 2-2, 2-15
- ? PFKEY command, 2-2
- ? PTF command, 2-2, 2-16 to 2-17
- ? RELEASE command, 2-2, 2-17
- ? SET command, 2-2, 2-18 to 2-19
- ? SET FOR command, 2-19
- ? SET GRAPH command, 2-2, 2-21
- ? SET NOT command, 2-20
- ? SET SETCOMMAND &myvar, 3-10

- ? STAT command, 2-2, 2-22
- ? STYLE command, 2-2, 2-24 to 2-25
- ? SU command, 2-2, 2-26
- ? USE command, 2-2, 2-26

A

- ACCBLN parameter, 1-3
- ACCEPT attribute, 1-27
- Access Files, 5-3
 - loading, 5-3 to 5-4
 - VIDEOTR2, A-27
- access to data, 3-7
- accessing data sources, 5-8
- accessing FOCUS data sources, 5-8
- activating HiperFOCUS, 5-13
- ADDRSPACE, 5-19
- AGGR[RATIO] parameter, 1-4
- ALL parameter, 1-4
- allocating files, 5-17, 5-18
- ALLOWCVTERR parameter, 1-5
- amper variables, 3-10, 3-26, 3-29, 3-49, 3-63
 - return values, 8-24
- applications, 8-51
 - creating, 3-36
 - running, 8-51
- AS phrase, 1-6
- ASNAMES parameter, 1-6
- AUTOINDEX parameter, 1-6
- AUTOPATH parameter, 1-7
- AUTOSTRATEGY parameter, 1-7
- AUTOTABLEF parameter, 1-7

B

BINS parameter, 1-8
 BLKCALC parameter, 1-8
 BOTTOMMARGIN parameter, 1-8
 branching, 3-10, 3-15
 buffering techniques, 5-8
 BUSDAYS parameter, 1-9
 BYPANEL parameter, 1-9
 BYSCROLL parameter, 1-9

C

CACHE parameter, 1-10, 5-20
 calculated values, 6-25
 MODIFY requests and, 6-27 to 6-28
 sliding window, 6-25
 canceling a procedure, 3-14
 CAR data source, A-11
 CARTESIAN parameter, 1-11
 CDN parameter, 1-11
 CENT-ZERO parameter, 1-12
 CHECK FILE command, 6-10
 clearing substitutions, 4-11
 -CLOSE command, 3-9, 3-81
 -CLOSE ddname command, 3-9
 -CMS command, 3-9, 3-81
 -CMS RUN command, 3-9
 COLUMNSCROLL parameter, 1-12
 COMASTER data source, A-21
 combined structures, 2-3
 displaying, 2-3
 command lines, 3-42
 displaying, 3-42

command statistics, 2-22
 displaying, 2-22
 commands, 1-2, 6-5, 6-10
 ? LET, 4-10
 changing, 3-62
 CHECK FILE, 6-10
 COMPUTE, 6-25
 DEFINE, 6-5, 6-19
 Dialogue Manager, 3-9, 3-80
 ENDECHO, 4-11
 EXEC, 3-7, 3-41
 LET CLEAR, 4-11
 LET ECHO, 4-10
 LET SAVE, 4-12
 LOAD, 5-2
 LOAD MODIFY, 5-5
 nesting, 3-40
 query, 2-2
 QUIT, 3-44
 SET, 1-2
 UNLOAD, 5-2
 WINDOW COMPILE, 8-83
 WINDOW PAINT, 8-52
 comments, 3-8
 including in procedures, 3-8 to 3-9
 COMPILE command, 5-7
 compiled requests, 5-5
 running, 5-5
 compound -IF tests, 3-18
 COMPUTE command, 6-25
 sliding window, 6-25
 substitutions, 4-9
 COMPUTE parameter, 1-12
 concatenating variables, 3-64
 conditional branching, 3-17 to 3-18
 configuring HiperFOCUS, 5-13 to 5-14
 conversions, 6-31
 currency, 7-2 to 7-3
 DATE NEW option, 6-31
 dates, 6-31

- converting currencies, 7-2 to 7-3, 7-9, 7-11 to 7-12
 - currency data source, 7-2
- COUNTWIDTH parameter, 1-13
- COURSES data source, A-16
- creating procedures, 3-6
- creating temporary files, 5-16 to 5-17
- cross-century dates, 6-1, 6-5, 6-30
 - MODIFY requests and, 6-5
- CRTCLEAR command, 3-9, 3-81
- CRTFORM command, 3-9, 3-49, 3-75, 3-82
- CRTFORMs, 3-10
- currencies, 7-2
 - converting, 7-2 to 7-3
- CURRENCY attribute, 7-6
- currency data source, 2-5, 7-2
 - activating, 7-8
 - creating, 7-4, 7-6
 - CURRENCY attribute, 7-6
 - currency codes, 7-4 to 7-6
 - displaying, 2-5
 - querying, 7-14

D

- data access
 - restricting, 3-7
- data entry, 3-9, 3-75
- data sources, 2-9, 2-26
 - displaying, 2-26
 - statistics, 2-9, 2-10
- date format, 6-5
 - sliding window and, 6-5
- DATEDISPLAY parameter, 1-13
- DATEFNS parameter, 1-14
- DATEFORMAT parameter, 1-14

- dates, 3-33, 6-2
 - converting, 6-31
 - default display format, 6-30
 - display options, 6-31
 - displaying, 3-58
 - functions, 6-31
 - setting, 3-33
 - system, 6-31
 - time stamp, 6-31
 - validating, 6-6
- DATETIME parameter, 1-14
- debugging procedures, 3-42
- DECODE function, 3-28
- default century, 6-31
- default values, 3-70
- DEFAULTS command, 3-10, 3-68, 3-70, 3-83
- DEFCENT parameter, 1-15, 6-2 to 6-3, 6-5 to 6-7
 - COMPUTE command, 6-25, 6-27
 - DEFINE command, 6-19
 - MODIFY requests and, 6-5 to 6-6
 - querying, 6-12
- DEFINE command, 6-5
 - sliding window, 6-5, 6-19
 - substitutions, 4-9
- defined functions, 2-11
 - displaying, 2-11
- Dialogue Manager, 3-2, 3-36
- Dialogue Manager commands, 3-1 to 3-2, 3-9, 3-80
 - *, 3-8 to 3-9, 3-80
 - ?, 3-80
 - ? &[string], 3-10, 3-51
 - ? SET COMMAND &myvar, 3-10
 - CLOSE, 3-9, 3-81
 - CLOSE ddname, 3-9
 - CMS, 3-9, 3-81
 - CMS RUN, 3-9
 - CRTCLEAR, 3-9, 3-81
 - CRTFORM, 3-9, 3-75, 3-82
 - DEFAULTS, 3-10, 3-70, 3-83
 - EXIT, 3-10, 3-12, 3-13, 3-83
 - GOTO, 3-10, 3-15, 3-84

Dialogue Manager commands (*continued*)

- HTMLFORM, 3-10, 3-84
- IF, 3-10, 3-15, 3-85
- INCLUDE, 3-10, 3-38, 3-86
- label, 3-10, 3-87
- MVS RUN, 3-10, 3-87
- PASS, 3-7, 3-10, 3-87
- PROMPT, 3-10, 3-49, 3-73, 3-88
- QUIT, 3-10, 3-12, 3-14, 3-89
- QUIT FOCUS, 3-14 to 3-15
- READ, 3-10, 3-72, 3-90
- REPEAT, 3-10, 3-24, 3-91
- RUN, 3-10, 3-12, 3-92
- SET, 3-10, 3-26 to 3-27, 3-32, 3-71, 3-92
- testing, 3-43
- TSO RUN, 3-10, 3-93
- TYPE, 3-10, 3-11, 3-93
- WINDOW, 3-10, 3-49, 3-75, 3-94, 8-3
- WRITE, 3-10, 3-45, 3-95

Dialogue Manager functions

- TRUNCATE, 3-30

Dialogue Manager variables, 3-1, 3-5

- direct prompting, 3-49

- DISPLAY parameter, 1-15

- displaying command lines, 3-42

- DTSTRICT parameter, 1-16

- dynamic window, 6-4, 6-10

E

- EDIT function, 3-29

- EDUCFILE data source, A-7

- EMPDATA data source, A-17

- EMPLOYEE data source, A-3

- EMPTYREPORT parameter, 1-16

- ENDECHO command, 4-11

- Entry Menu, 8-53

- environment variables, 5-23

- error files, B-1

- CMS, B-2

- MVS, B-2

- error messages, 2-15, B-1

- displaying, B-3

- displaying explanations of, 2-15

- ERROROUT parameter, 1-17

- ESTRECORDS parameter, 1-18

- euro currency, 7-1 to 7-2

- converting, 7-2 to 7-3, 7-9, 7-11 to 7-12

- EUROFILE parameter, 1-19, 7-8

- error messages, 7-8

- restrictions, 7-8

- evaluating rule statements, 5-23 to 5-24

- EX command, 3-7, 3-41

- EXEC command, 3-7, 3-41

- execution windows, 8-11

- EXIT command, 3-10, 3-12, 3-13, 3-83

- exiting from FOCUS, 3-14 to 3-15

- EXPERSON data source, A-18

- EXTAGGR parameter, 1-19

- EXTHOLD parameter, 1-19

- EXTSORT parameter, 1-20

- EXTTERM parameter, 1-20

F

- FDEFCENT attribute, 6-3, 6-13

- field names windows, 8-8

- field variables, 3-62

- field-level sliding window, 6-13, 6-15

- DEFCENT attribute, 6-15

- MODIFY requests and, 6-17

- YRTHRESH attribute, 6-15

- FIELDNAME parameter, 1-20

- fields, 2-5
 - displaying, 2-5
- file contents windows, 8-9
- file directory table, 2-6, 2-7
 - displaying, 2-6
- file management, 5-12
- file names windows, 8-7
 - creating, 8-49
- file utilities, 5-2
- file-level sliding window, 6-13
 - FDEFCENT attribute, 6-13
 - FYRTHRESH attribute, 6-13
- FILENAME attribute, 1-21
- files, 5-2
 - closing, 3-9
 - disorganization, 2-10
 - displaying information for, 2-15, 5-6
 - loaded files, 2-15
 - loading, 5-2 to 5-3
 - reading from, 3-46
 - unloading, 5-2 to 5-3
 - writing to, 3-45 to 3-46
- FILTER parameter, 1-21
- FINANCE data source, A-14
- FIXRETRIEVE parameter, 1-22
- flow of control, 3-10
- flow of execution, 3-12
- FOC144 parameter, 1-22
- FOC2GIGDB parameter, 1-23
- FOCALLOC parameter, 1-23
- FOCCOMP file, 5-5
- FOCSTACK parameter, 1-23
- FOCUS Database Server, 2-26
 - displaying information for, 2-26

- FOCUS facilities, 5-1 to 5-2
- format specifications, 3-76 to 3-77
- full-screen data entry, 3-49, 3-75
- function keys, 2-16, 8-26
 - assigning phrases to, 4-12
 - displaying assignments, 2-16
- functions, 6-22
 - calling, 3-33 to 3-35
 - for dates, 6-31
 - running, 3-9 to 3-10
 - sliding window and, 6-5, 6-22, 6-29
- FYRTHRESH attribute, 6-3, 6-13

G

- GGDEMOG data source, A-29
- GGORDER data source, A-30
- GGPRODS data source, A-31
- GGSALES data source, A-32
- GGSTORES data source, A-33
- global sliding window, 6-7, 6-8
- global variables, 2-27, 3-5, 3-49, 3-53
 - displaying values, 2-27
 - querying, 3-53 to 3-54
- Gotham Grinds data sources, A-29
 - GGDEMOG, A-29
 - GGORDER, A-30
 - GGPRODS, A-31
 - GGSALES, A-32
 - GGSTORES, A-33
- GOTO command, 3-10, 3-15 to 3-16, 3-84
- goto values, 8-3, 8-25, 8-61
- graph parameters, 2-21
 - setting, 2-21

H

HDAY parameter, 1-24

HiperBUDGET facility, 5-14
installing, 5-14
limits, 5-14 to 5-15

HiperBUDGET installation parameters, 5-14 to 5-15

HiperCache option, 5-12, 5-19, 5-20

HIPERCACHE parameter, 1-24

HIPEREXTENTS parameter, 1-24

HiperFile option, 5-12, 5-16 to 5-17
allocating files explicitly, 5-16
creating temporary sort files, 5-19
hiperspaces, 5-16

HIPERFILE parameter, 1-25

HiperFOCUS configuration parameters, 5-14

HiperFOCUS environment variables, 5-23

HiperFOCUS installation parameters, 5-14

HiperFOCUS option, 5-12 to 5-13, 5-18 to 5-21
configuring, 5-13 to 5-14
installing, 5-13 to 5-14
limiting use of hiperspace, 5-16

HIPERFOCUS parameter, 1-25, 5-13

HIPERINSTALL parameter, 1-25

HIPERLOCKED parameter, 1-26

HiperRule option, 5-12, 5-21 to 5-24
environment variables, 5-23

hiperspace limits, 2-11
displaying, 2-11 to 2-12

HIPERSPACE parameter, 1-26

hiperspaces, 5-12, 5-16 to 5-17, 5-19
controlling size, 5-19

HLISUDUMP parameter, 1-27

HLISUTRACE parameter, 1-26

HOLD fields, 2-12
displaying, 2-12

HOLDATTR parameter, 1-27

HOLDLIST parameter, 1-28

HOLDSTAT parameter, 1-28

horizontal menus, 8-5
creating, 8-14

HOTMENU parameter, 1-29

-HTMLFORM command, 3-10, 3-84

I

-IF command, 3-10, 3-15, 3-17 to 3-18, 3-85

-IF tests, 3-18
compound, 3-18
operators and functions, 3-19

IMMEDTYPE parameter, 1-29

implied prompting, 3-49, 3-75

-INCLUDE command, 3-10, 3-37 to 3-40, 3-86

INDEX parameter, 1-30

indexed variables, 3-64 to 3-65
creating, 3-65

installing HiperBUDGET, 5-14

installing HiperFOCUS, 5-13 to 5-14

interactive data entry, 3-49

J

JOBFILE data source, A-6

join structures, 2-13
displaying, 2-13

JOINOPT parameter, 1-30

K

KEEPDEFINES parameter, 1-30

L

-label command, 3-10, 3-87

LANG parameter, 1-31

LEADZERO parameter, 1-32

LEDGER data source, A-13

LEFTMARGIN parameter, 1-32

legacy dates, 6-5
 sliding window and, 6-5

LET CLEAR command, 4-11

LET command, 4-1 to 4-4, 4-7 to 4-9, 4-12

LET ECHO command, 4-10

LET SAVE command, 4-12

LET substitutions, 4-1
 debugging, 4-10
 displaying, 4-10

limiting hiperspace size on VM, 5-19

LINES parameter, 1-32

LOAD command, 5-2 to 5-3

LOAD MODIFY command, 5-5

load procedures, A-1 to A-2

loaded files, 2-15, 5-6
 displaying information for, 2-15

loading files, 5-2 to 5-3
 Access Files, 5-3 to 5-4
 FOCCOMP files, 5-5
 Master Files, 5-3 to 5-4
 MODIFY requests, 5-5

loading procedures, 5-3 to 5-4

local variables, 3-5, 3-49, 3-52

looping, 3-24 to 3-25
 controlling, 3-32
 ending, 3-26

loops, 3-10

M

Main Menu, 8-54

mask option of EDIT function, 3-29

Master Files, 2-8, 6-13 to 6-14
 defining sliding windows in, 6-13 to 6-16, 6-17
 displaying field information, 2-8
 loading, 5-3 to 5-4
 samples, A-1 to A-2
 virtual fields, 6-23 to 6-24

MASTER parameter, 1-33

MAXLRECL parameter, 1-33

menus, 8-4
 creating, 8-2
 horizontal, 8-5, 8-14
 pulldown, 8-16
 vertical, 8-5

MESSAGE parameter, 1-33

messages
 displaying, 3-10 to 3-11

MINIO parameter, 1-34, 5-8 to 5-9, 5-10 to 5-11

MODIFY requests, 5-5, 6-2
 compiling, 5-7 to 5-8
 DEFCENT parameter, 6-5
 loading, 5-5
 YRTHRESH parameter, 6-5

modules, 5-7
 running, 5-7

MOVIES data source, A-24

multi-input windows, 8-12
 creating, 8-18

MULTIPATH parameter, 1-35

multiple procedures, 3-37

multiple-line substitutions, 4-8

-MVS RUN command, 3-10, 3-87

N

National Language Support (NLS), 2-14

natural date literals, 3-33

nesting commands, 3-40

nesting procedures, 3-40

NLS (National Language Support), 2-14

NODATA parameter, 1-35

non-FOCUS files, 3-10
 reading, 3-10

null substitutions, 4-7

O

ONLINE-FMT parameter, 1-36

open-ended procedures, 3-41 to 3-42

ORIENTATION parameter, 1-36

P

page handling, 5-18

PAGE-NUM parameter, 1-37

PAGESIZE parameter, 1-37

PANEL parameter, 1-39

PAPER parameter, 1-40

parameter settings, 2-18
 displaying, 2-18 to 2-20

parameter values, 3-51
 querying settings, 3-51

parameters, 1-3
 querying value settings, 3-51
 setting, 1-2 to 1-3

-PASS command, 3-7, 3-10, 3-87

PASS parameter, 1-40

passwords
 setting, 3-7, 3-10

PAUSE parameter, 1-41

PAYHIST data source, A-20

PCOMMA parameter, 1-41

PF keys, 2-16, 8-26
 displaying assignments, 2-16

PFnn command, 8-26

PFnn parameter, 1-42

positional variables, 3-69

PREFIX parameter, 1-42

PRINT parameter, 1-42

PRINTPLUS parameter, 1-43

procedures, 3-2, 5-3
 calling, 3-41
 canceling, 3-14
 creating, 3-6, 3-41 to 3-42
 debugging, 3-42
 exiting, 3-10
 including, 3-10, 3-37 to 3-40
 including comments, 3-8
 load, A-1 to A-2
 loading, 5-3 to 5-4
 nesting, 3-40
 prompting, 3-49
 running, 3-2, 3-7, 3-12
 SAMPLE, 8-31
 sample in Window Painter, 8-51
 security, 3-7, 3-44
 variables and, 3-49

PROD data source, A-10

-PROMPT command, 3-10, 3-49, 3-73 to 3-74, 3-88

prompting, 3-49, 3-75, 3-79

PTFs, 2-16

displaying, 2-16 to 2-17

pull-down menus, 8-16

creating, 8-16

Q

QUALCHAR parameter, 1-44

QUALTITLES parameter, 1-44

queries

testing status, 3-23

query commands, 2-1 to 2-2, 6-12

? &&, 2-27

? COMBINE, 2-3

? DEFINE, 2-4

? EUROFILE, 2-5, 7-14

? F, 2-5

? FDT, 2-6

? FF, 2-8

? FILE, 2-9

? FUNCTION, 2-11

? HBUDGET, 2-11

? HOLD, 2-12

? JOIN, 2-13

? LANG, 2-14

? LET, 2-14

? LOAD, 2-15, 5-6

? n, 2-15

? PTF, 2-16

? RELEASE, 2-17

? SET, 2-18

? SET GRAPH, 2-21

? SET NOT, 2-20

? STAT, 2-22

? STYLE, 2-24

? SU, 2-26

? USE, 2-26

QUIT command, 3-44

-QUIT command, 3-10, 3-12, 3-14, 3-89

-QUIT FOCUS command, 3-14 to 3-15

R

-READ command, 3-10, 3-46, 3-68, 3-72 to 3-73, 3-90

REBUILDMSG parameter, 1-45

RECAP-COUNT parameter, 1-45

RECORDLIMIT parameter, 1-45

records

accessing, 3-10

writing, 3-10

recursive substitutions, 4-8 to 4-9

REGION data source, A-15

release numbers, 2-17

displaying, 2-17

-REPEAT command, 3-10, 3-24 to 3-25, 3-91

requests

translating, 4-4

return value display windows, 8-9

return values, 8-24

RIGHTMARGIN parameter, 1-46

-RUN command, 3-10, 3-12, 3-92

S

SALES data source, A-8, A-9

sample data sources, A-2

CAR, A-11

COMASTER, A-21

COURSES, A-16

creating, A-1 to A-2

EDUCFILE, A-7

EMPDATA, A-17

EMPLOYEE, A-3

EXPERSON, A-18

sample data sources (*continued*)

- FINANCE, A-14
- Gotham Grinds data sources, A-29
- JOBFILE, A-6
- LEDGER, A-13
- MOVIES, A-24
- PAYHIST, A-20
- PROD, A-10
- REGION, A-15
- SALES, A-8, A-9
- TRAINING, A-19
- VIDEOTR2, A-26
- VideoTrk, A-24

SAVEMATRIX parameter, 1-46

saving substitutions, 4-12

SBORDER parameter, 1-46

SCREEN parameter, 1-47

screening values with -IF tests, 3-19 to 3-22

screens, 3-9, 8-51

- clearing, 3-9
- Entry Menu, 8-53
- Main Menu, 8-54
- Utilities Menu, 8-72
- Window Creation Menu, 8-57
- Window Design Screen, 8-59
- Window Options Menu, 8-61

SET command, 1-1 to 1-3

-SET command, 3-10, 3-26 to 3-27, 3-30, 3-32 to 3-35, 3-44, 3-68, 3-71, 3-92

SET parameters, 1-1 to 1-3, 6-2

- ACCBLN, 1-3
- AGGR[RATIO], 1-4
- ALL, 1-4
- ALLOWCVTERR, 1-5, 6-31
- ASNAMES, 1-6
- AUTOINDEX, 1-6
- AUTOPATH, 1-7
- AUTOSTRATEGY, 1-7
- AUTOTABLEF, 1-7
- BINS, 1-8
- BLKCALC, 1-8
- BOTTOMMARGIN, 1-8
- BUSDAYS, 1-9

SET parameters (*continued*)

- BYPANEL, 1-9
- BYSCROLL, 1-9
- CACHE, 1-10, 5-20
- CARTESIAN, 1-11
- CDN, 1-11
- CENT-ZERO, 1-12
- COLUMNSCROLL, 1-12
- COMPUTE, 1-12
- COUNTWIDTH, 1-13
- DATEDISPLAY, 1-13, 6-31
- DATEFNS, 1-14
- DATEFORMAT, 1-14
- DATETIME, 1-14
- DEFCENT, 1-15, 6-2, 6-7
- DISPLAY, 1-15
- DTSTRICT, 1-16
- EMPTYREPORT, 1-16
- ERROROUT, 1-17
- ESTRECORDS, 1-18
- EUROFILE, 1-19, 7-8
- EXTAGGR, 1-19
- EXTHOLD, 1-19
- EXTSORT, 1-20
- EXTTERM, 1-20
- FIELDNAME, 1-20
- FILENAME, 1-21
- FILTER, 1-21
- FIXRETRIEVE, 1-22
- FOC144, 1-22
- FOC2GIGDB, 1-23
- FOCALLOC, 1-23
- FOCSTACK, 1-23
- HDAY, 1-24
- HIPERCACHE, 1-24
- HIPEREXTENTS, 1-24
- HIPERFILE, 1-25
- HIPERFOCUS, 1-25, 5-13
- HIPERINSTALL, 1-25
- HIPERLOCKED, 1-26
- HIPERSPACE, 1-26
- HLISUDUMP, 1-27
- HLISUTRACE, 1-26
- HOLDATTR, 1-27
- HOLDLIST, 1-28
- HOLDSTAT, 1-28
- HOTMENU, 1-29

SET parameters (*continued*)

- IMMEDTYPE, 1-29
- INDEX, 1-30
- JOINOPT, 1-30
- KEEPDEFINES, 1-30
- LANG, 1-31
- LEADZERO, 1-32
- LEFTMARGIN, 1-32
- LINES, 1-32
- MASTER, 1-33
- MAXLRECL, 1-33
- MESSAGE, 1-33
- MINIO, 1-34, 5-8, to 5-11
- MULTIPATH, 1-35
- NODATA, 1-35
- ONLINE-FMT, 1-36
- ORIENTATION, 1-36
- PAGE-NUM, 1-37
- PAGESIZE, 1-37
- PANEL, 1-39
- PAPER, 1-40
- PASS, 1-40
- PAUSE, 1-41
- PCOMMA, 1-41
- PFnn, 1-42
- PREFIX, 1-42
- PRINT, 1-42
- PRINTPLUS, 1-43
- QUALCHAR, 1-44
- QUALTITLES, 1-44
- REBUILDMSG, 1-45
- RECAP-COUNT, 1-45
- RECORDLIMIT, 1-45
- RIGHTMARGIN, 1-46
- SAVEMATRIX, 1-46
- SBORDER, 1-46
- SCREEN, 1-47
- SHADOW, 1-47
- SHIFT, 1-48
- SORTLIB, 1-48
- SPACES, 1-48
- SQLTOPTIF, 1-49
- SQUEEZE, 1-49
- STYLE[SHEET], 1-50
- SUMPREFIX, 1-50
- SUSI, 1-51
- SUTABSIZE, 1-51

SET parameters (*continued*)

- TEMP[DISK], 1-51
- TERM, 1-51
- TERMTRAN, 1-52
- TESTDATE, 1-52, 6-10
- TEXTFIELD, 1-53
- TITLE, 1-53
- TOPMARGIN, 1-53
- TRACKIO, 1-54, 5-18
- TRMOUT, 1-54
- UNITS, 1-54
- USER, 1-55
- WEFTAB, 1-55
- WEEKFIRST, 1-56
- WIDTH, 1-56
- XRETRIEVAL, 1-57
- YRTHRESH, 1-57, 6-2, 6-7

setting parameters, 1-2 to 1-3

SHADOW parameter, 1-47

SHIFT parameter, 1-48

sliding window, 6-1 to 6-3

- calculated value, 6-25, 6-27 to 6-28

- date format, 6-5

- date options, 6-30

- DEFCENT parameter, 6-2, 6-5

- defining in a Master File, 6-13 to 6-16

- defining with SET, 6-7 to 6-8, 6-10

- dynamic window, 6-4, 6-10

- field-level, 6-13, 6-15 to 6-18

- file-level, 6-13 to 6-14, 6-18

- functions and, 6-22, 6-29

- global, 6-7 to 6-8

- legacy dates, 6-5

- YRTHRESH parameter, 6-2, 6-5

SORTLIB parameter, 1-48

SPACES parameter, 1-48

special variables, 3-61

SQLTOPTTF parameter, 1-49

SQUEEZE parameter, 1-49

stacked commands, 3-2

- executing, 3-10, 3-12 to 3-13

- startup profiles, 3-36 to 3-37
- statistical variables, 3-5, 3-49, 3-59
 - &LINES, 3-60
 - querying, 3-61
- storing rules, 5-24
- structure diagrams, A-1 to A-2
- STYLE[SHEET] parameter, 1-50
- StyleSheet parameter settings, 2-24
 - dispalying, 2-24 to 2-25
- StyleSheets, 2-24
- SU machine, 2-26
- subroutines
 - running, 3-10
- substitutions, 2-14, 4-1 to 4-7
 - assigning to function keys, 4-12
 - checking, 4-10
 - clearing, 4-11
 - debugging, 4-10
 - displaying, 2-14
 - multiple-line, 4-8
 - recursive, 4-8 to 4-9
 - saving, 4-12
- substitutions for phrases, 4-7
- substitutions for variable phrases, 4-5 to 4-6
- SUMPREFIX parameter, 1-50
- SUSI parameter, 1-51
- SUTABSIZE parameter, 1-51
- system date, 6-31
- system defaults, 3-97
- system variables, 3-5, 3-49, 3-54, 3-58
 - &DATE, 3-57
 - &FOCFOCEXEC, 3-57
 - &YYMD, 3-58
 - testing, 3-73

T

- TED text editor, 3-6
- TEMP[DISK] parameter, 1-51
- temporary fields, 2-4
 - displaying, 2-4
- temporary files, 5-16
 - allocating, 5-17 to 5-18
 - creating, 5-16
 - moving, 5-18
- TERM parameter, 1-51
- TERMTRAN parameter, 1-52
- TESTDATE parameter, 1-52, 6-10
- testing procedures, 3-42
- text display windows, 8-7
 - creating, 8-35, 8-39
- text input windows, 8-6
- TEXTFIELD parameter, 1-53
- TITLE parameter, 1-53
- TOPMARGIN parameter, 1-53
- TRACKIO parameter, 1-54, 5-18
- trailing blanks, 3-30
 - deleting, 3-31
- TRAINING data source, A-19
- TRMOUT parameter, 1-54
- TRUNCATE function, 3-30 to 3-31
- TSO RUN command, 3-10, 3-93
- TYPE command, 3-10 to 3-11, 3-93

U

- unconditional branching, 3-10, 3-16
- UNITS parameter, 1-54
- UNLOAD command, 5-2 to 5-3
- unloading files, 5-2 to 5-3
- USER parameter, 1-55
- Utilities Menu, 8-72
- utilization statistics, 2-11
 - displaying, 2-11 to 2-12

V

- valid values, 3-77 to 3-79
- value ranges, 3-77
 - specifying, 3-77
- values, 3-27, 8-3
 - default, 3-97
 - defining, 3-27 to 3-28
 - goto, 8-25, 8-61
 - prompting for, 3-75, 3-79
 - querying, 3-51
 - screening, 3-19 to 3-22
 - setting, 3-34
 - supplying, 3-49, 3-66 to 3-78
 - verifying, 3-76
- variable phrases, 4-5, 4-6
- variable substitution, 3-49
- variable types, 3-5
 - amper, 3-49
 - global, 3-5, 3-49, 3-53
 - local, 3-5, 3-49, 3-52
 - positional, 3-69
 - special, 3-61
 - statistical, 3-5, 3-49, 3-59
 - system, 3-5, 3-49, 3-54

- variable values, 3-66 to 3-71
 - supplying, 3-71 to 3-79
- variables, 3-26
 - appending values, 3-64
 - assigning, 3-34
 - assigning values to, 3-10
 - changing commands, 3-62
 - computing, 3-27 to 3-28
 - concatenating, 3-64
 - evaluating, 3-62 to 3-63
 - procedures and, 3-49
 - querying, 3-51
 - setting, 3-10
 - supplying values, 3-66 to 3-74, 3-76 to 3-79

- vertical menus, 8-5
 - creating, 8-41, 8-47

VIDEOTR2 data source, A-26

VideoTrk data source, A-24

- virtual fields, 2-4, 6-19
 - adding, 3-40
 - defining windows for, 6-19, 6-21, 6-23 to 6-24
 - displaying, 2-4

W

WEBTAB parameter, 1-55

WEEKFIRST parameter, 1-56

WIDTH parameter, 1-56

window applications, 8-22

-WINDOW command, 3-10, 3-49, 3-75, 3-94, 8-3

WINDOW COMPILE command, 8-83

Window Creation Menu, 8-57

Window Design Screen, 8-59

Window facility, 8-22

- window files, 8-3 to 8-4
 - creating, 8-33

- Window Options Menu, 8-61
 - Display list, 8-65
 - Heading, 8-64
 - Help window, 8-67, 8-71
 - Hide list, 8-66
 - Line break, 8-69
 - Multi-select, 8-70
 - Popup, 8-67
 - Quit, 8-71
 - Show a window, 8-64
 - Unshow a window, 8-64
 - WINDOW PAINT command, 8-52
 - Window Painter, 3-10, 8-2
 - goto values, 8-3
 - invoking, 8-52
 - main menu, 8-35
 - screens, 8-51
 - tutorial, 8-29
 - window files, 8-3
 - Window Painter tutorial, 8-29
 - SAMPLE, 8-31
 - window file, 8-33
 - windows, 8-4, 8-11
 - &WINDOWNAME variable, 8-26
 - &WINDOWVALUE variable, 8-26
 - creating, 8-2, 8-14
 - field names, 8-8
 - file contents, 8-9
 - file names, 8-7
 - horizontal, 8-5, 8-14
 - multi-input, 8-12, 8-18
 - procedures, 8-21
 - windows (*continued*)
 - return value display, 8-9
 - returning to caller, 8-25
 - running, 8-28
 - text display, 8-7
 - text input, 8-6
 - types, 8-4
 - vertical menu, 8-5
 - word substitutions, 2-14, 4-1 to 4-7
 - assigning to function keys, 4-12
 - checking, 4-10
 - clearing, 4-11
 - debugging, 4-10
 - displaying, 2-14
 - multiple-line, 4-8
 - recursive, 4-8, 4-9
 - saving, 4-12
 - WRITE command, 3-10, 3-45 to 3-46, 3-95
 - writing rules, 5-21 to 5-22
- ## X
- XCONFIG ADDRSPACE, 5-19
 - XRETRIEVAL parameter, 1-57
- ## Y
- YRTHRESH parameter, 1-57, 6-2 to 6-3, 6-5, 6-7
 - COMPUTE command, 6-25
 - DEFINE command, 6-19
 - MODIFY requests and, 6-5 to 6-6
 - querying, 6-12
 - with COMPUTE, 6-27

Reader Comments

In an ongoing effort to produce effective documentation, the Documentation Services staff at Information Builders welcomes any opinion you can offer regarding this manual.

Please use this form to relay suggestions for improving this publication or to alert us to corrections. Identify specific pages where applicable. You can contact us through the following methods:

Mail: Documentation Services – Customer Support
Information Builders, Inc.
Two Penn Plaza
New York, NY 10121-2898

Fax: (212) 967-0460

E-mail: books_info@ibi.com

Web form: <http://www.informationbuilders.com/bookstore/derf.html>

Name: _____

Company: _____

Address: _____

Telephone: _____ Date: _____

E-mail: _____

Comments:

Reader Comments